electronics

Article

Content Caching in Mobile Edge Computing Based on User Location and Preferences Using Cosine Similarity and Collaborative Filtering

Gul-E-Laraib [1], Sardar Khaliq uz Zaman [1,*], Tahir Maqsood [2], Faisal Rehman [1], Saad Mustafa [1], Muhammad Amir Khan [1,*], Neelam Gohar [3], Abeer D. Algarni [4] and Hela Elmannai [4]

[1] Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad 22060, Pakistan
[2] Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Lahore 54000, Pakistan
[3] Department of Computer Science, Shaheed Benazir Bhutto Women University, Peshawar 25120, Pakistan
[4] Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
* Correspondence: skhaleeq@cuiatd.edu.pk (S.K.u.Z.); amirkhan@cuiatd.edu.pk (M.A.K.)

Abstract: High-speed internet has boosted clients' traffic needs. Content caching on mobile edge computing (MEC) servers reduces traffic and latency. Caching with MEC faces difficulties such as user mobility, limited storage, varying user preferences, and rising video streaming needs. The current content caching techniques consider user mobility and content popularity to improve the experience. However, no present solution addresses user preferences and mobility, affecting caching decisions. We propose mobility- and user-preferences-aware caching for MEC. Using time series, the proposed system finds mobility patterns and groups nearby trajectories. Using cosine similarity and CF, we predict and cache user-requested content. CF predicts the popularity of grouped-based content to improve the cache hit ratio and reduce delay compared to baseline techniques.

Keywords: caching; mobile edge computing; content preferences; user mobility; MovieLens

Citation: Gul-E-Laraib; Zaman, S.K.u.; Maqsood, T.; Rehman, F.; Mustafa, S.; Khan, M.A.; Gohar, N.; Algarni, A.D.; Elmannai, H. Content Caching in Mobile Edge Computing Based on User Location and Preferences Using Cosine Similarity and Collaborative Filtering. *Electronics* **2023**, *12*, 284. https://doi.org/10.3390/electronics12020284

Academic Editor: Claus Pahl

Received: 23 November 2022
Revised: 6 December 2022
Accepted: 12 December 2022
Published: 5 January 2023

1. Introduction

As a result of advances in computer and wireless communication, the number of internet-connected devices is rising rapidly. Mobile devices have acquired sophisticated sensing and computational capabilities in recent years. Combined with the evolution of deep learning, this creates several opportunities for practical applications, such as in medical, safety, and transportation networks [1,2].

Mobile edge computing (MEC) brings intelligence closer to the edge where data are produced. MEC pushes cloud services—such as computation, networking, and storage—to the edge of the mobile network to fulfil the requirements of applications which are computation intensive such as online games, delay-sensitive tasks such as augmented reality applications, and high-bandwidth-demanding applications, e.g., mobile big data and analytics [3,4]. MEC focuses on important metrics such as delay and high bandwidth. These metrics are accomplished by processing users' tasks on the MEC server instead of the cloud. Mobile edge computing has emerged as a solution that is superior to novel methods in terms of conserving energy, enhancing throughput, preserving tolerable levels of privacy, and assuring adequate levels of security when compared to other options currently available. MEC has several advantages, such as low latency, low energy consumption, context awareness, and cost reduction [5]. An overview of how content caching in MEC works is shown in Figure 1.

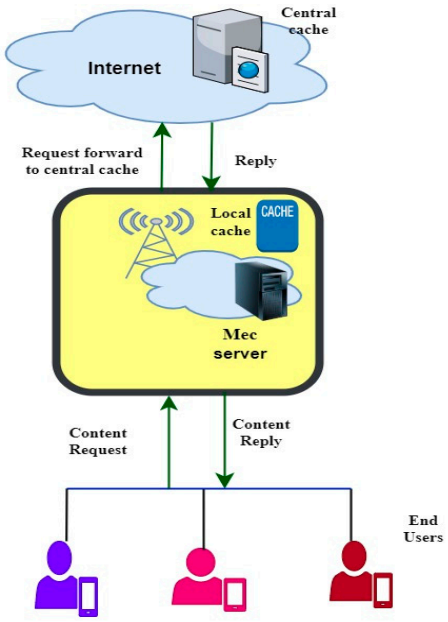

Figure 1. Content caching in MEC.

As mobile data traffic increases daily, the backhaul data requirement becomes the major bottleneck for decreasing costs and increasing the operators' revenue [6]. The research community has recently focused on content caching as a solution to this problem [7]. Content caching can reduce redundant data propagation, due to which backhaul traffic can be greatly reduced. Additionally, caching can improve several performance parameters, such as spectrum efficiency, energy efficiency, and transmission delay [8]. Several real-world contexts can benefit from content popularity and user mobility. As an illustration, the 2022 FIFA World Cup is currently underway in Qatar. To keep up with the latest matches of their favorite teams, fans increasingly turn to match updates, match highlights, videos of goals, video replays, and live matches. On-the-go viewers crave access to live events, highlights, and news coverage (while moving from one place to another). Content caching in MEC is beneficial because of two reasons. Firstly, the future edge network will be diverse because of using different base stations (BSs) as MECs. Thus, caching is being implemented at various BSs at different locations. Initially, the requested content is retrieved from the central server if that content is not cached locally. The content caching mechanism keeps a copy of the content for future use. If every time a user request is forwarded to the core, it increases the network traffic and delay. Secondly, using cheaper storage servers at several BSs caching mechanisms at small base stations (SBSs) and macro base stations (MBSs) becomes very easy and cost-effective. Content caching strategies cache the content from the centralized server to the end-users, thus improving the QoE of users and reducing delay. The content selection process decides which content to be cached, which should be replaced, and how much time a particular content will be placed in the cache. There are different criteria based on this decision, such as content popularity and diversity. To achieve network efficiency, optimize what content to be cached, how to store the content, and how to retrieve the content from the caching entity by considering user mobility [9].

The amount of video traffic has significantly increased thanks to recent technological advancements. The authors of [10] presented a caching technique that considers the popularity of material as well as the mobility of users to alleviate the challenges associated with video traffic. The popularity of the content can be affected by users' movement on various edge servers; therefore, a content caching approach is recommended to handle this issue. This proposed strategy decides to cache the content based on the predicted value of the content's popularity and user mobility. Another issue with edge servers is that they have limited storage space, due to which they cannot bear high load pressure. Limited storage means that these edge servers have lesser memory capability, i.e., they can store a limited

amount of data. A content placement technique based on the marginal gain is provided in [10]. as a potential solution to this problem. This strategy has two components: the first is called content placement, and the second is called content replacement. The cloud data center collects information about the content and edge server for content placement [11]. This information includes the size and popularity of the edge servers' content, storage, and processing capabilities. Then, the cloud server decides to place the content based on the collected information. For content replacement, the cloud computes the marginal gain of cached content and the popularity of local content. If the marginal benefit or local popularity of fresh content exceeds the cached content, then the cached content is replaced. The suggested technique decreases storage expenses and latency [10].

User mobility is a key challenge in MEC architecture, and it affects several contexts, such as caching, computational offloading, and connected users [12]. Content caching in MEC can fulfil the demands of delay-sensitive applications, reduce network traffic, and enhance end-user QoE. However, the mobility of users with often handoffs among BSs can negate the benefits of content caching; therefore, such content caching techniques need to be designed to consider mobility [13]. Mobile users move from one place to another place or change their point of connectivity from one BS to another or move from one cache server to another. When users move before the all of the content is downloaded, that content becomes useless unless or until there is a proper handover mechanism from one cache server to another. The content should be cached based on users' mobility to ensure the best possible service with lesser delay [14].

User preference is another challenge that significantly affects content caching. It shows the user's interest in a particular piece of content and, indirectly, their chance of requesting it. It is possible to anticipate user preferences by analyzing prior content requests and the commonalities between users. When we talk about user preferences, we're referring to a user's personality as it relates to their likes and dislikes for various types of content. Social connections, patterns of mobility, and other user factors can influence user preferences (age, occupation, and location, for example). The user's interest in a specific piece of content can be established by examining their download history. User preferences are important to provide services to users intelligently and improve the QoE of users. Suppose that this content is cached on the MEC server, which is undesirable for users and wastes cache storage. User preferences have a substantial effect on caching because, if we have user preferences, user interest is deterministic, and content may be cached [15]. In this way, the user will find the desired content directly with much lesser delay when issuing a request. With prior knowledge of user preferences, a high cache hit rate, low latency, and enhanced quality of service (QoS) can be achieved. When a user is mobile, it is difficult to download a full file from a single server as the user continuously changes location. A user may request particular content from a server, and as the downloading starts, the user moves towards another server without waiting to download the requested content. The overview of the proposed framework is illustrated in Figure 2.

As shown in Figure 2, our objective is to develop a framework for content caching at MEC while considering mobility and user preferences to enhance performance. The performance is measured by considering the cache hit rate and delay [15]. The contributions of the paper are given below.

- We propose a content caching approach for the MEC environment that takes mobility and user preferences into account. This strategy can store the most popular content while considering user mobility, hence minimizing delay and enhancing the cache hit ratio.
- The proposed scheme mainly comprises the content popularity prediction module and the user mobility prediction module. The output of both modules is used in a content caching algorithm.
- We utilize user-based collaborative filtering to predict the popularity of content. User-based CF first finds similar users with common patterns and then recommends items that these similar users are interested in.

- We use the cosine similarity technique to determine the similarity among the users of both datasets, MovieLens and GeoLife.
- We conduct numerous experiments to demonstrate the effectiveness of our method. As input, we utilized user ID, longitude/latitude, movie ratings, and the total number of ratings. In terms of cache hit ratio and delay, the experimental findings demonstrate that the proposed framework outperforms three baseline procedures and one cutting-edge technique.

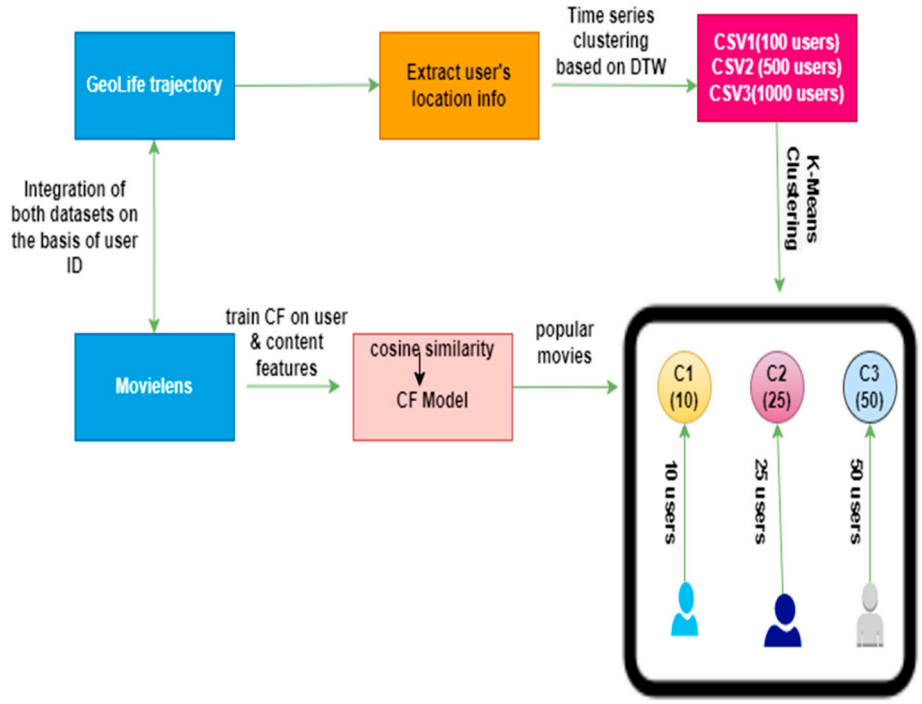

Figure 2. User preferences demonstration.

The remainder of the paper is laid out as follows. The Section 2 examines the related work. The proposed caching scheme's system architecture is given in Section 3. Section 4 is where we define our problem. The framework's implementation is outlined in Section 5. Section 6 contains a full discussion of the experimental setup and the results. Finally, Section 7 brings the paper to a conclusion.

2. Related Work

Multiple social media platforms—including TikTok, Instagram, and Facebook—implement live streaming, resulting in a significant rise in video traffic [16]. To solve this issue, the authors of [17] proposed a caching technique that considers the material's popularity and users' mobility. The popularity of material is affected by the mobility of users among various edge servers; therefore, a caching strategy is recommended to handle this issue. This approach selects whether or not to cache content depending on its popularity and portability. In addition, edge servers have limited storage space, meaning they cannot withstand heavy load strain. To tackle this issue, [17] presents a content placement approach based on marginal gain. This strategy has two components: content placement and content replacement. For content placement, a cloud data center collects information about the content and edge server, including the material's size and popularity, storage, and processing capability. The cloud server then selects where to place the material based on the gathered data. For content replacement, the cloud computes the cached material's marginal benefit and local content popularity. If the marginal benefit or local popularity of future content is greater than the cached content, it is replaced. The proposed technique minimizes storage costs and response times [17].

In [18], mobility-aware content caching strategy is presented to decrease energy consumption and reduce the burden of network traffic. To predict mobility, small cells are grouped into various clusters. The main objective of [18] is to minimize energy used in content delivery to mobile users. The energy consumption problem is divided into two sub-problems. The first is about inter-cluster caching, which is considered an NP-hard problem, so this polynomial heuristic algorithm is proposed. Then the authors compare mobility-aware strategies with mobility-oblivious strategies. The second part considers the heterogeneity of small cells; this part aims to place the contents in small cells (SCs) in the cluster. The proposed strategy reduces energy consumption by up to 56% compared to the traditional caching strategies. Moreover, the complexity of the proposed algorithm is also comparatively low. Authors in [19] address the problem of cache placement in terms of highly mobile users. The speed of users varies, and the point connection changes irregularly. Due to the high mobility of users, it is impractical to download the requested content from one BS. This results in a high downloading delay. Because of the mobility of users between BSs, only part of the requested content can be downloaded from the current BS. The remaining download request is forwarded to the macro base station (MBS), and consequently this delay increases. The authors proposed mixed-integer nonlinear programming with the Markov renewal process to solve the problem. To model user mobility, Markov renewal process is used. Then, a greedy algorithm is designed based on sub-modular optimization. A heuristic search mechanism is used for large-scale content caching. The technique proposed in [19] improves the cache hit ratio up to 13% and 16% compared with femto caching (FC) and mobility-aware caching with a fixed amount of data that can be delivered (MCFD). Because of dense deployment, a user can be in the range of multiple BSs, which increases the probability of a cache hit. However, the cache hit ratio of the proposed algorithm decreases when users move at high speed. Moreover, the proposed strategy caches the content based on global popularity, and global popularity prediction increases the complexity.

User mobility and user association are addressed in [20]. This caching problem is transformed into a nonlinear NP-hard problem. The original long-term problem is split into two smaller problems for a better solution. These simpler problems are addressed in two steps first step is content caching, and the second step is user association. The mobility-aware online caching algorithm is proposed for caching, and for finding user location, a lazy re-association algorithm is presented in [20]. The goal of the authors is to minimize the cost. Unlike the traditional caching algorithm, the proposed strategy updates the content in a short interval of time. Every user connects with suitable BS to obtain the desired content. Each BS can only handle a limited number of users depending on the resources. Users can retrieve the requested content in a single association period. Markov chain is used to predict user mobility. It is assumed that users move from one BS to another on a sequential path within one adjacent time slot. The performance of the proposed scheme is almost twice in comparison with the existing caching schemes in terms of user requests.

In [21], the authors investigate the content caching problem in non-orthogonal multiple access (NOMA)-MEC systems. NOMA technology enables multiple users to use the orthogonal resources at a time. NOMA can handle more users than the number of subcarriers, with this delay decreasing and spectral efficiency increasing. The authors aim to reduce latency by optimizing caching strategy, computational resources, and energy consumption. This problem is considered a mixed-integer nonlinear problem. This problem is efficiently solved by a block successive upper-bound minimization (BSUM) strategy. The proposed algorithm performs under several constraints, such as energy consumption, offloading decisions, and computation and storage capability of edge servers. Single carrier NOMA is used in the proposed algorithm where time division duplex (TDD) is used for uplink and downlink. The proposed BSUM algorithm is quicker and has superior decomposition capabilities. The suggested approach outperforms the benchmark algorithms in overall completion delay, cache hit rate, and energy usage (local-only, all-offloading, and equal resource). However, in [21], the authors consider only one base station within 400 m, due to

which far users experience a high delay in retrieving the requested content. The immense growth in video traffic causes communication delays and slow download speeds. Video prefetching or content caching can alleviate heavy network traffic. The authors in [22]. exploit data retrieving and cache replacement in mobile edge computing to reduce the overall network traffic. The authors designed an effective function based on the mobility probability of users and the popularity of video content. The proposed utility function optimizes the cache resource allocation. The probability of a user's handoff from one cell to another is determined through the individual mobility model (IMM). The presented cache replacement is an NP-hard problem, and to resolve this problem mobility-aware video prefetch caching and replacement (MAVPCR) algorithm is proposed. The proposed MAVPCR achieves a 25.76% high cache hit rate. Similarly, the algorithm above achieves a 30% reduction in delay compared to the other algorithms; however, this reduction is difficult to maintain when the cache size increases [22].

Information about user mobility and users' demand can enhance a network's capability to improve users' QoE. In [23], the authors study proactive content caching in roadside units (RSUs) by considering the impact of user mobility on caching decisions. The authors proposed a non-cooperative caching strategy in which each RSU independently takes the caching decisions [24]. In [25], a cooperative edge and cloud environment is deployed to reduce response time and lessen the backhaul's traffic strain. To satisfy the aforementioned aims, we describe a technique for collaboratively storing video content. This proposed method groups various edge servers within the cluster using the K-means algorithm. Calculating the latency and cost of caching is then used to define the content caching problem, which is achieved by caching the content on the edge servers. In addition, the marginal gain of content caching is estimated by analyzing the cost and latency associated with caching.

When vehicles keep moving, content popularity is not easy to predict. Moreover, the cached content can become outdated soon because the sojourn time of fast-moving vehicles in a BS is very small. To resolve these two issues, the authors of [26] proposed a mobility-aware proactive edge caching scheme based on federated learning (MPCF). The proposed scheme collaboratively learns a global model to predict content popularity. A context-aware adversarial autoencoder predicts the most popular content, and the predicted contents are cached to the RSUs.

In [27], the authors resolve the issue of mobility in service caching by a mobility-aware service caching technique. Using an idealized geometric model, the destination of a traveling user is predicted. The proposed model utilizes current information to identify the BS to which the user may be connected once the service is complete. The authors of [28] introduced a multi-agent reinforcement learning (MARL)-based cooperative content caching policy for MEC in situations when the popularity and user choice of the cached material are uncertain. The proposed technique makes use of the user's previous content requests. The method presented in [28] considerably increases the cache hit ratio and minimizes the downloading delay. Proactive content caching can ease the stress of traffic on the backhaul links by caching popular content at base stations. Among various other factors affecting the caching design, the two most important are the mobility of users and the content preference of users. These two changes are addressed in [29]. The content preference problem is formulated as a decentralized multitask learning (DRMTL) problem. The impact of the social circle on defining users' behavior while gathering the contextual data of users is often neglected [30]. The authors in [31] proposed a context-aware caching scheme based on social behavior (CCSB), which combines various contextual information of users such as age, gender, etc. The proposed scheme describes the impacts of social networks on users' behavior. MEC caches the content to nearby users to decrease the delay. The demands and behavior of users have a significant impact on caching decisions. In [32], authors proposed a Smart caching algorithm for MEC architecture. Smart caching considers users' demands and behavior. The proposed caching algorithm effectively predicts dynamic user content

demands by analyzing social characteristics and hence decides to cache adoptively on mobile edge.

Content caching in MEC is identified as a significant approach that enhances energy consumption and reduces latency. However, due to the limited capacity in the cache, it is not easy to design any effective caching technique that fulfils the demands of ever-changing user preferences. To address these issues, in [33] the author proposed a new proactive content caching strategy which predicts the changing content popularity and user preferences flexibly in a dynamically varying environment. This scheme is based on online learning architecture. The proposed process of learning consists of two learning models first is long short-term memory (LSTM) based local learning, and the second is ensemble-based meta-learning. Generally, the works mentioned above have used techniques for caching content based on how mobile users are, and some have also looked at how users like things. Some papers looked at the connection between user mobility and user association. The above strategies store most of the content on the edge nodes, such as edge servers, small base stations, RSUs, etc. Most works [13,14] deal with the mobility problem by designing caching strategies. When a user is on the go, it is hard to download full files. Some work considered the latency problem due to mobility and video traffic which is solved using different algorithms. Only the influence of user mobility on caching solutions is considered by certain research, ignoring the variation in user popularity caused by mobile user mobility. The current popularity of a piece of content cannot predict the future demand for that piece of content since the popularity of material fluctuates over time and with the mobility of consumers. While developing content caching strategies, the articles mentioned above do not address user mobility or preference. As a result, the approaches above could not meet user expectations and negatively impacted productivity. When users' demands or user preferences and user mobility are handled improperly, it affects the caching performance adversely. In the given literature, different techniques have been proposed to enhance cache performance by considering user mobility and preference, as shown in Table 1.

Table 1. Comparative analysis of related works.

	Presented Techniques	**Parameters**					
		Energy Efficiency	**Latency**	**Cache hit Ratio**	**Network Cost**	**Computational Overhead**	**Storage Utilization**
1	Polynomial heuristic algorithm [18]	✓	✓	×	✓	✓	✓
2	A greedy algorithm for contact duration aware cooperative content placement [19]	✓	×	✓	×	×	×
3	Mobility-aware online content caching algorithm [20]	×	×	×	✓	×	✓
4	Block successive upper-bound minimization method [21]	✓	✓	✓	✓	×	✓
5	Mobility-aware video prefetch caching and replacement [22]	×	×	✓	×	✓	×
6	Non cooperative caching scheme [23]	×	✓	✓	×	×	×
7	Content collaborative caching strategy [25]	×	✓	✓	×	✓	×
8	Proactive edge caching scheme based on federated learning algorithm [26]	×	×	✓	×	×	×
9	Service caching algorithm [27]	✓	×	✓	×	×	×
10	MARL-based cooperative content caching [17]	×	✓	✓	×	×	×
11	Mobility aware decentralized regularized multitask learning [29]	×	×	✓	✓	✓	✓
12	Context-aware caching scheme with social behavior [19]	×	×	✓	✓	✓	✓
13	Single-layer nonlinear convolutional neural network [20]	×	✓	×	✓	✓	✓
14	A novel proactive content caching control based on hierarchical online learning [21]	✓	✓	✓	✓	×	×

Most of the techniques consider user mobility and user preference separately and avoid combining both of them. Therefore, the focus of this work is first to predict user mobility and then determine the user preference. Based on these user preferences, the content popularity will be estimated. At last, the predicted popular content will be cached while caching the content. We compared the proposed caching technique with LFU, FIFIO, and LRU as in other studies and considered delay and cache hit ratio as the performance parameters inspired by several related works [15,26,34].

3. System Model

We incorporate the well-known MovieLens and GeoLife trajectories into our system. MovieLens is a collection of user-submitted movie ratings. Ratings offered by moviegoers are used to learn about user preferences. To determine which films are the most popular, we look at how highly users rated them and how similar their preferences are. Therefore, in the end, the movies with the highest ratings recieve more attention than the ones with the fewest. The longitude, latitude, and location information of users with their IDs can be found in the GeoLife trajectory. Thus, we use the user ID to link the two databases together. This information is used to assess the current movement patterns of users and predict the next probable location based on these patterns.

User requests are served faster when they are checked in the local cache rather than the server's database first, as long as they are not too specific. If the requested content is not found in the local cache, then the content will be retrieved from the central cache which will cause the delay, which is the first evaluation parameter. Delay can be calculated using Equation (1).

$$DL_i = \frac{D_x}{R_i} \tag{1}$$

where DL_i represents the delay of downloading content from node i, D_x is the size of content x to be downloaded, and R_i the data transmission rate of node i.

Our proposed model caches the content after learning user preferences. When popular content is cached, the probability that a requesting user will find the requested content increases, which ultimately enhances the cache hit ratio, our second evaluation parameter. The cache hit ratio can be calculated using Equation (2).

$$CH_r = \frac{CH}{(CH + CM)} \tag{2}$$

where CH_r is the cache hit ratio, CH is number of cache hits, while CM is number of cache misses. Our proposed model also addresses user mobility before caching a particular content. For user mobility, we use the GeoLife trajectory dataset, which consists of nonlinear data about the location information of users. We used a time-series clustering algorithm to classify the users into different groups (clusters) based on mobility. Those users that lie near each other are grouped. We used dynamic time wrapping (DTW) to measure the similarity between two temporal sequences. Provided sequences $X = \{x_0, \ldots, x_n\}$ and $Y = \{y_0, \ldots, y_n\}$, DTW between X and Y is given by optimization shown in Equation (3).

$$DTW_{(x,y)} = \underset{\pi}{min} \sqrt{\sum_{(i,j)\in\pi} d(x_i, y_j)^2} \tag{3}$$

DTW is measured as squrred root of the sum of squared distances among each element of sequence X and its corresponding nearest point in sequence Y.

Cosine similarity uses the dot product of two items to determine their similarity. Considering certain items represented as vectors, cosine similarity calculates angles between the vectors and the results is cosine similarity values. The smaller angle values represent higher similarity between the items. Consequently, those items are recommended that have higher cosine similarity, i.e., lower angle between the items. Cosine similarity between

vectors $\vec{v_c}$ and $\vec{v_s}$ item can be calculated as shown in Equation (4). Where vectors $\vec{v_c}$ and $\vec{v_s}$ represent user and movies, respectively. Table 2 explain the notations used in the equations and formulas.

$$cos\left(\vec{v_c}, \vec{v_s}\right) = \frac{\vec{v_c} \,.\, \vec{v_s}}{||\vec{v_c}|| \times ||\vec{v_s}||} \tag{4}$$

Table 2. Notations and definitions.

S. No	Notations	Definitions
1.	n	Quantity of users
2.	r	Requests from users
3.	DLi	Delay of downloading content
4.	D_x	size of content x
5.	CH	Cache hits
6.	R_i	the data transmission rate of node i.
7.	CM	Cache misses
8.	CH_r	Cache hit ratios
8.	U_{id}	User IDs
10.	M_r	Movie ratings
11.	N_r	Number of ratings
12.	M_g	Movie genre
13.	C	Predicted popular content
14.	Q	List
15.	δ_C	Caching server
16.	C_p	Popular content
17.	δ_p	Next predicted server

4. MUPAC: Mobility and User Preference Aware Content Caching Framework

We introduce the MUPAC framework for MEC. As mentioned earlier, existing solutions cache the contents depending on user mobility and content popularity, but no solution combines user mobility and content popularity for joint prediction decisions. Therefore, we illustrate the proposed framework that comprises two modules. The first module uses collaborative filtering and cosine similarity for user preferences prediction and similarity among users, respectively. For user preferences, we use MovieLens data. Furthermore, in the second module, we use dynamic time wrapping (DTW) to predict the user locations. For user mobility data, we use GeoLife trajectories. The trajectory of this dataset consists of a sequence of timestamped points, each of which contains the information of user ID, longitude, latitude, spot ID, altitude, and variation in the location to determine the average velocity (time series data). A broad range of anonymous users has their IDs and spot ID through which we can know the current location and nearby locations and their longitudes and latitude. Since the dataset above is a time-series dataset that consists of locations and time, we used time-series clustering to analyze the previous mobility patterns of the users and map that previous location to the next probable location of the user.

After time series clustering, we take 50 to 1000 users in three CSV files, respectively. Then used, the famous machine learning technique K-means to form fifty clusters. These clusters are formed based on the user's location. The cluster centroids are calculated through dynamic time wrapping (DTW) because Euclidian distance does not work well on time series data. The second component is content popularity. Collaborative filtering (CF) was then utilized to identify the best material based on user feedback. Instead, then focusing on the interpersonal dynamics between individuals, CF examines the connections between various types of content. Finding information comparable to previously liked content is how recommendation engines figure out what to suggest to a user. The core concept of CF revolves around recommending or predicting what a user might want to see. Moviegoers' opinions can be collected from the ratings or votes they give the film.

The desired solution is achieved using two datasets, i.e., the GeoLife trajectories and MovieLens. In GeoLife, we have trajectories, users IDs, latitude, longitude, distance, days, and duration, and MovieLens consisting of movie ratings (votes), movie tags, user IDs, and time stamps. The next location will be predicted by obtaining the latitude and longitude of the location where the user is currently present. The Algorithm 1 utilizes the CF prediction and mobility prediction data as input and cache the popular content at optimal edge server. The proposed framework includes the following modules.

- User interface module: There are two user interface modules. The first module retrieves the user's information, i.e., user ID, current movie ID, movie ratings, and movie genre, from the MovieLens dataset. The other module is used to extract user information (user ID, longitude, latitude, spot ID, and time stamp) from GeoLife trajectory
- User feature construction module: This module creates the user's features using the temporal and spatial information of currently connected users.
- User preference construction module: It uses the user's features to draw user preferences.
- Mobility prediction module: user's next location is predicted based on previous location history (longitude, latitude).
- Clustering module: users are grouped based on location, i.e., those who lie near each other are grouped.
- Cache management module: cache the most popular material and keep it around for a set period. User interface modules can only access content stored on the local MEC Server; otherwise, it must be downloaded from the core network and transmitted to the MEC server.
- Content rank module: A requested content's rank can be altered to reflect the popularity of a particular piece of content among its users. This can be achieved by boosting the popularity of popular pieces of material and lowering the popularity of less popular pieces.

The detailed working of the MUPAC framework is shown in Figure 3.

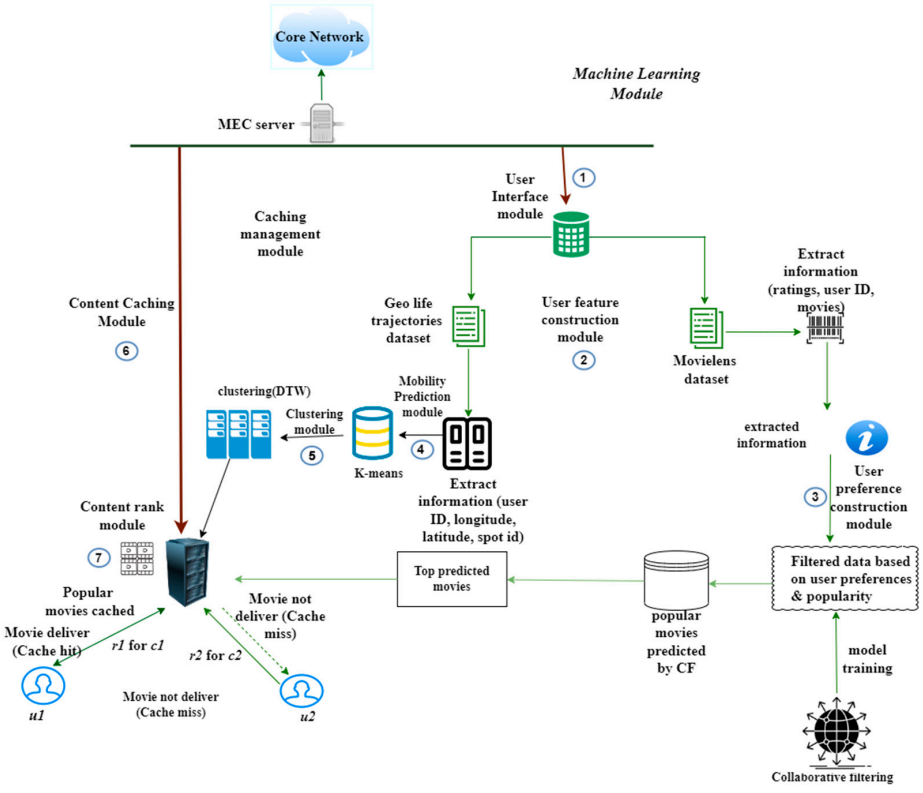

Figure 3. Modular workflow of the MUPAC framework.

Algorithm 1: Content Caching Algorithm

Inputs: Popular contents using CF model, predicted user location using DTW model
Output: Content cached at the predicted server

1. **while** C_p in $Q \neq \varphi$
2. **if** δ_C *(server capacity) is sufficient*
3. Cache C_p to δ_p
4. **else**
5. update the list of servers Q
6. Repeat step 1
7. Delete C_p from Q
8. Update capacity of servers δ in Q
9. **End** while

4.1. User Mobility Prediction Module

This module consists of two steps. We extract the user's mobility information (longitude, latitude, timestamp, and spot ID) in the first step. After extracting, we analyze this information. The second step is related to predicting the next location of users and classifying users based on the nearby location. Since our data are time-series data consisting of time and location information, we used a time-series clustering algorithm to classify the users into different groups (clusters) based on mobility. Those users that lie near each other are grouped. We used dynamic time wrapping (DTW) instead of Euclidean distance because DTW measures the similarity between two temporal sequences. Since the data are nonlinear, we used time-series clustering to flatten the time series into a table with a column for each index, as shown in Table 3. We collect similar samples of time series through DTW. Then the cluster center is measured with respect to DTW, which gives us a group of time series. As a result, the centroid formed mimics the shape of all the users in that cluster. After time series clustering, we used K-means to form clusters of nearby users. We take three scenarios of user density, i.e., in the first scenario, we form a cluster of 10 users. In the second scenario, we formed a cluster of 25 users; in the third scenario, we grouped 50 users to form a cluster.

Table 3. Flattening time series.

Trajectories	**t1**	**t2**	**t3**	. . . .	**tn**
Trajectory 1	(1, 2)	(3, 4)	(5, 6)	. . . . .	(n1, n2)
Trajectory 2	(2, 4)	(5, 6)	(8, 10)	. . . . .	(n1, n2)
Trajectory 3	(3, 7)	(9, 4)	(1, 6)	. . . . .	(n1, n2)
Trajectory n	(9, 2)	(3, 4)	(5, 6)	. . . . .	(n1, n2)

4.2. Content Popularity Prediction Module

The content popularity prediction algorithm begins by extracting data such as movie ID, movie votes, movie genres, user ID, and time stamp. After obtaining the information, the data are filtered according to user preferences and popularity. Votes cast by users at different times are utilized to determine user preferences. We determined a movie's popularity based on the number of votes it received from users. The initial vote count is used to determine the most popular films for each user, and then cosine similarity is used to determine the similarities between users. As illustrated in Figure 4, user-based collaborative filtering is employed to anticipate the most popular content for each cluster.

Collaborative Filtering (CF) Model

Collaborative filtering (CF) is a recommendation system technique. Due to its accurate forecast of user interest, it is one of the most successful recommendation algorithms. We divide the data into 90–10, i.e., we train the CF model on 90% of our data, and 10% of the data are used for testing purposes. User information (i.e., user ID, longitude/latitude,

number of votes, spot ID, and score) and movie information (i.e., movie ID, votes, movie genre, and timestamp) are fed as input to the CF model. DF model consists of multiple layers where the output of one layer is fed as input of the next layer. The bottom input layer consists of two features that describe user ID and content. Since we use pure collaborative filtering, we used user ID and movie ID as input features (user is the same parameter in both datasets). The input layer is fed into multi-layer CF architecture, where the user and content features are mapped to the prediction score. The final output layer is the predicted score Yic. An architecture of the collaborative filtering model is shown in Figure 5. As stated earlier, we have used collaborative filtering (CF) to extract user preferences and cosine similarity to determine the similarity among users in the MovieLens dataset. The same technique has been used in different works, and its complexity is not high because we used a few user preferences such as content likes, dislikes, and the number of views [35,36].

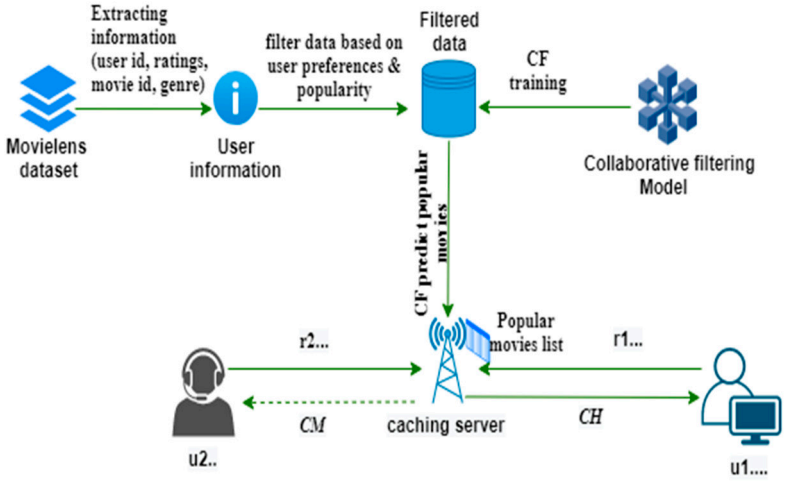

Figure 4. Popularity prediction module.

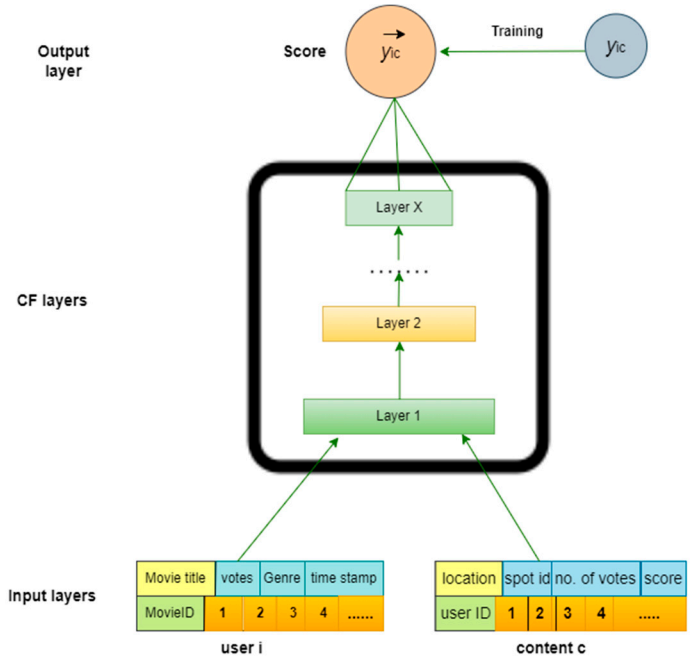

Figure 5. Collaborative filtering (CF) model.

5. Result and Evaluations

In this section, we first explain the experimental setup and data sets used. Then we discuss the accuracy and validation of the prediction model. Finally, we discuss the performance of the proposed framework compared with the baseline techniques.

5.1. Experimental Setup

We form clusters based on the nearby location of users using three scenarios of user density. Keras is a deep learning network using TensorFlow as the backend to create the CF model [37]. In this experiment, we use two datasets: the MovieLens 1M dataset from the MovieLens website [22] and the GeoLife trajectory dataset. For the mobility of users, we use the GeoLife trajectory dataset. It contains different trajectories of users in the form of userID, longitude, latitude, and spot ID. It is a time-series dataset that contains time-series data in the form of timestamped points. MovieLens dataset contains about 1 million ratings from 6040 anonymized users on 3883 movies. As assumed in [23], to emulate users' content request process, we regard a user's movie rating as a request for that movie. We carried out comprehensive computer simulations following the 3GPP specifications and open-source data approved by telecom operators [38], as implemented in related works [39]. We used the MobFogSim simulator, which is dynamic and supports a wide range of IoT/mobile device platforms. The migration of computations to perform caching is handled by the MogFogSim, being a built-in feature of the simulator. Table 4 provides the details of model implementations and simulation settings.

Table 4. Simulation settings and modeling parameters.

S. No	Parameters	Value
1.	Number of iterations	50
2.	Rate of model learning	0.002
3.	Validations failures	8
4.	Hidden layers	5
5.	Number of polling layers	3
6.	Training data size	897,560
7.	Testing data size	102,440 (11%)
8.	Time taken to train the model	617 min
9.	Testing time	0.043 ms
10.	No of users	50–500
11.	Size of input data	1–18 MB
12.	Cache sizes	5–500
13	Base station radius	200 m
14.	Macro base station radius	1000 m
15.	No of Base stations	144

5.2. Evaluation of Prediction Model

Loss is a crucial aspect of neural networks. It is the error prediction of the neural network model as computed by the loss function. Figure 6 depicts the model's accuracy, which is directly related to validation. In addition, Figure 7 shows the validation loss of the MUPAC. After 50 iterations, the cumulative loss is close to 0.18.

5.3. Performance Results

Figure 8 depicts the cache hit rates for sizes ranging from 0 to 50 items. The outcome demonstrates that our suggested technique outperforms all other reference caching schemes. The cache hit ratio of all schemes increases as cache capacity grows. FIFO is the algorithm with the lowest cache hit ratio. Our proposed MUPAC beats MCPDCN, LFU, LRU, and FIFO since these techniques rely on historical data to predict future material's appeal. LFU and LRU are able to predict the popularity of static content, but they cannot account for dynamic popularity variations. MCPDCN calculates the content popularity through

the number of users' request, due to which it achieves less cache hit ratio as compared to MUPAC.

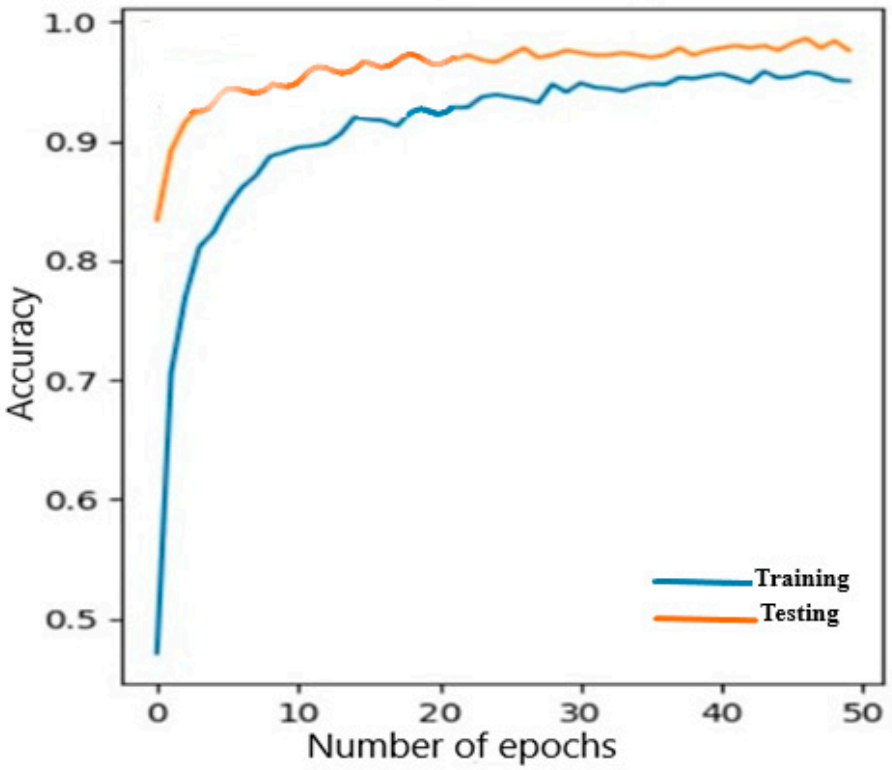

Figure 6. Learning model accuracy.

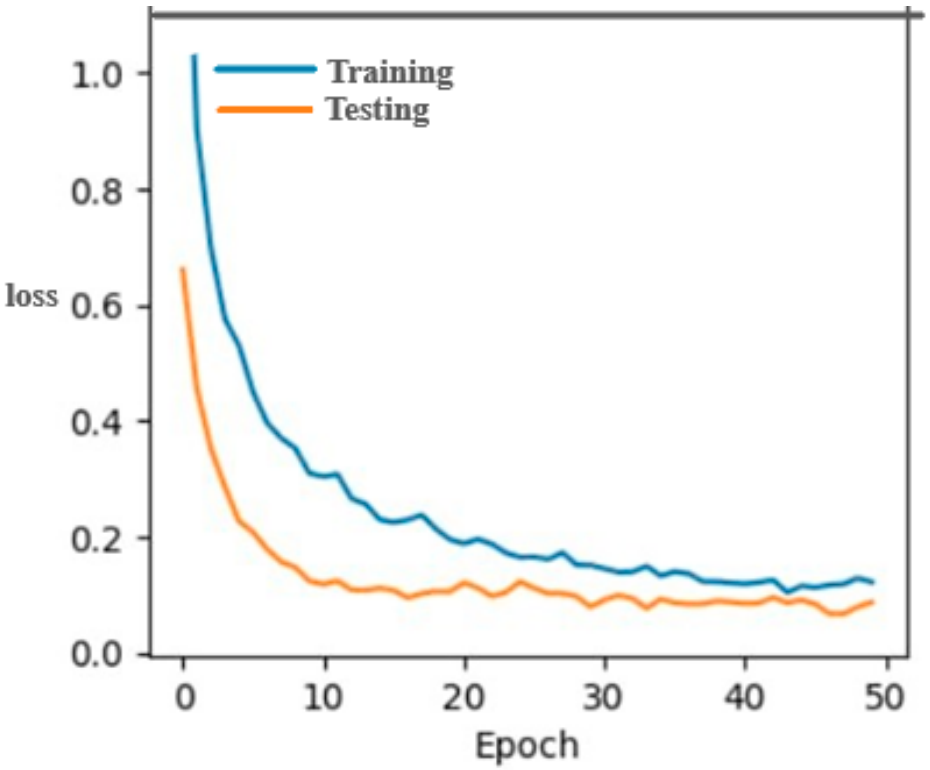

Figure 7. Learning model validation loss.

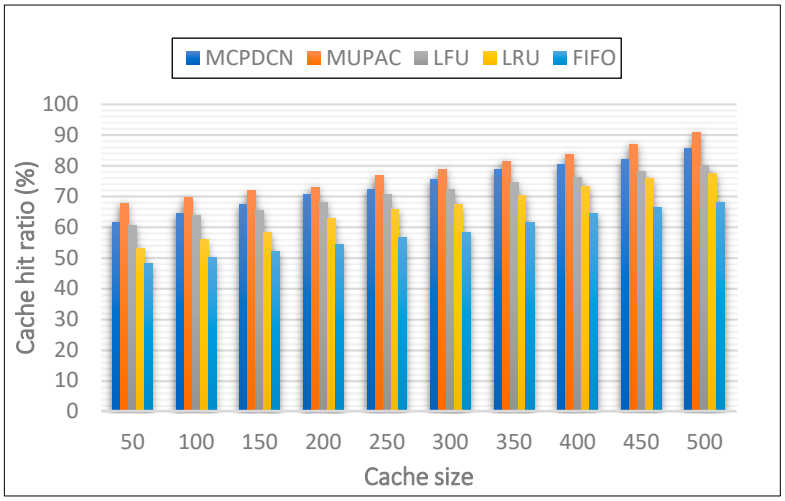

Figure 8. Cache hit ratio comparison.

The cache hit ratio of the base technique, i.e., mobility-aware content preference learning in decentralized caching networks and our proposed technique MUPAC is shown in Figure 9. It is shown that our scheme achieves a high cache hit ratio. The number of users varies from 50 to 500.

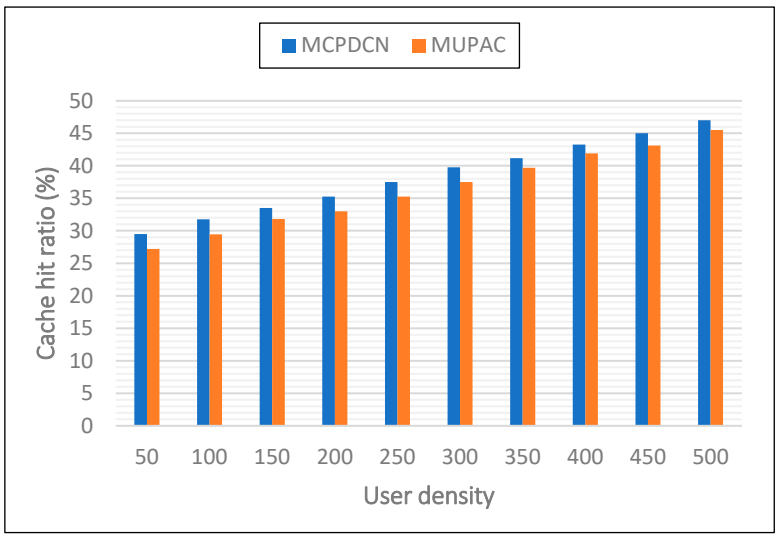

Figure 9. MUPAC vs MCPDCN (cache hit ratio) in terms of user density.

Figure 10 depicts the effect of user density on the cache hit ratio. The maximum cache size of the MEC server is 50 in this experiment, and the number of users ranges from 10 to 50. The data indicates that the cache hit ratio improves as the number of users rises.

The delay comparison of the MUPAC with LFU and LRU is shown in Figure 11. We perform this comparison by varying the number of users. Results demonstrate that as we increase the number of users, the delay experienced by our proposed MUPAC decreases. This is because our suggested MUPAC operates based on the users' predetermined content preferences and locations. As soon as a user connects to a server, the server quickly learns the user's preferences, estimates the next likely place the user will go to, and prefetches the material from the core if it was not previously cached. Now, whenever a user requests content, the server responds immediately.

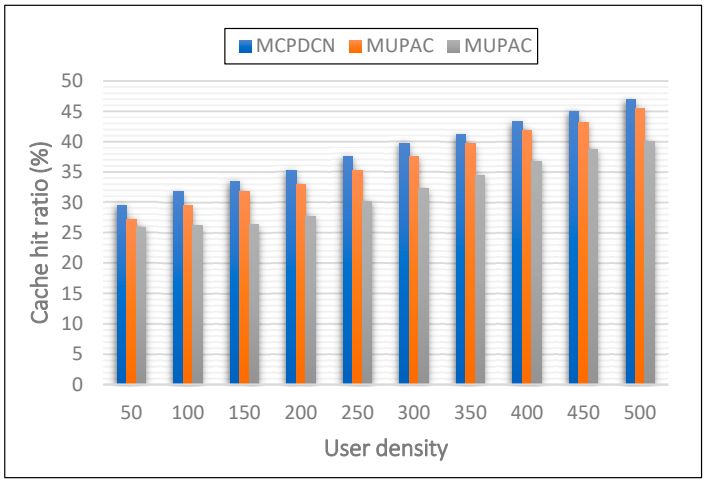

Figure 10. Cache hit ratio comparison in terms of user density.

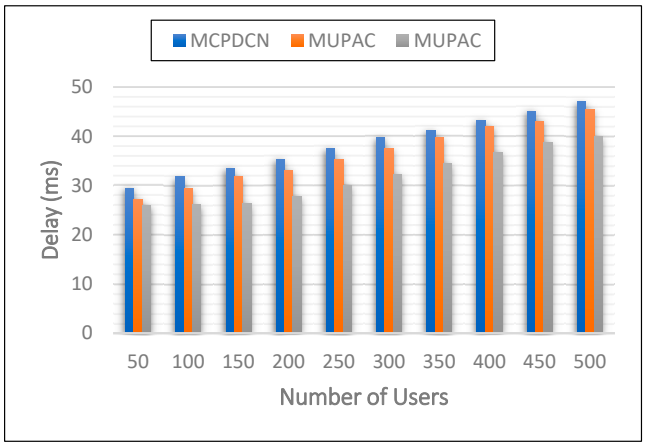

Figure 11. MUPAC vs LRU and LFU (delay) in terms of user density.

Figure 12 shows the delay comparison of our proposed MUPAC and the baseline technique MCPDCN is shown. The main difference between the proposed MUPAC and MCPDCN is predicting the content's popularity. MUPAC predicts content popularity based on user preferences using cosine similarity and collaborative filtering, whereas MCPDCN estimates content popularity based on the number of user requests. Due to this difference, the proposed MUPAC outperforms MCPDCN in delay.

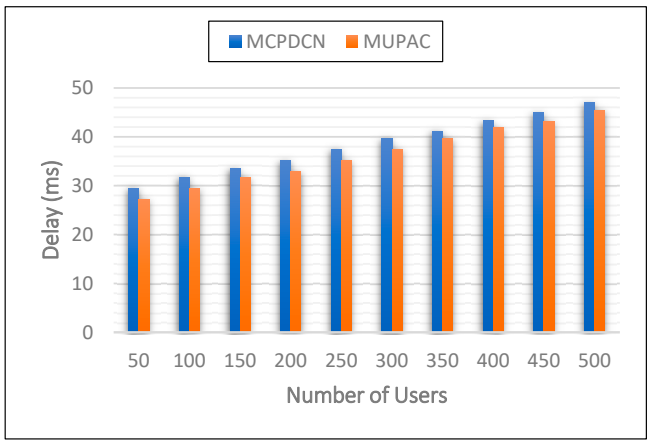

Figure 12. MUPAC vs. MCPDCN (delay) in terms of user density.

6. Conclusions

We analyze content caching in MEC to increase user QoE, cache utilization, cache hit ratio, and delay in this study. Diverse ways to cache content were thoroughly explored, revealing that caching can reduce network traffic and enhance the user experience. It is observed that all existing caching methods either function based on user preferences or user mobility. However, no work has examined these elements simultaneously, resulting in a poorer cache hit ratio and increased delay. We addressed user mobility and preferences simultaneously. We propose MUPAC (content caching aware of mobility and user preferences) for MEC architecture. This proposed framework first learns users' mobility patterns and then predicts users' preferences to determine content popularity. Using cosine similarity, we examined the user-content matrix to establish the relationships between various users' identities. The collaborative filtering (CF) model is used to predict user preferences, while time-series clustering (DTW) is used to assess a user's mobility. MUPAC enhances the cache speed, cache hit ratio, and delay greatly. Compared to the baseline techniques (LRU, LFU, and FIFO) in the MEC environment, MUPAC increases the cache hit ratio by 12% and reduces the delay by 5%.

7. Future Research Directions

Due to the Internet of Things and 5G cellular communication, most consumers are on the go with a wide range of requirements. The requirement for fetching the desired content on the move will be raised. In this research, we consider the MovieLens 1M dataset and cluster the nearby users based on time series clustering. In the future, we will use MovieLens 10M datasets, which will include more diverse users with dynamic requirements. Moreover, we will simulate the mobility of users using a simulator such as MobFogSim to achieve better results for mobility and see the impact of caching on energy consumption, task offloading failure probability, load balancing and resource utilization. We intend to work on these areas in our future work. We also intend to exploit the neural-networks-based recommendation technique to predict popular content for caching purposes and compare it with our results. In addition, recurrent neural networks can also be utilized to enhance the accuracy of user location prediction.

Author Contributions: G.-E.-L., S.K.u.Z., A.D.A., H.E. and T.M. conceived and designed the experiments; F.R., S.K.u.Z., M.A.K. and S.M. performed the simulations; M.A.K., S.M., N.G. and F.R. analyzed the results; G.-E.-L., S.K.u.Z., H.E. and S.M. wrote the paper, H.E., A.D.A., N.G. and F.R. technically reviewed the paper. All authors have read and agreed to the published version of the manuscript.

Funding: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R51), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No data were used to support this study. We have conducted simulations to evaluate the performance of the proposed protocol. However, any query about the research conducted in this paper is highly appreciated and can be asked by the author (Muhammad Amir Khan) upon request.

Acknowledgments: The authors sincerely appreciate the support from Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R51), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kyung, Y.; Song, T. CSV: Content Service Offloading System with Vehicular Caching. *Sensors* **2022**, *22*, 7967. [CrossRef] [PubMed]
2. Safi, A.; Ahmad, Z.; Jehangiri, A.I.; Latip, R.; Zaman, S.K.U.; Khan, M.A.; Ghoniem, R.M. A fault tolerant surveillance system for fire detection and prevention using LoRaWAN in smart buildings. *Sensors* **2022**, *22*, 8411. [CrossRef] [PubMed]

3. Zaman, S.K.U.; Jehangiri, A.I.; Maqsood, T.; Umar, A.I.; Khan, M.A.; Jhanjhi, N.Z.; Shorfuzzaman, M.; Masud, M. COME-UP: Computation Offloading in Mobile Edge Computing with LSTM Based User Direction Prediction. *Appl. Sci.* **2022**, *12*, 3312. [CrossRef]
4. Sandhu, A.K. Big data with cloud computing: Discussions and challenges. *Big Data Min. Anal.* **2021**, *5*, 32–40. [CrossRef]
5. Zaman, S.K.U.; Maqsood, T.; Ali, M.; Bilal, K.; Madani, S.A.; Khan, A.U.R. A load balanced task scheduling heuristic for large-scale computing systems. *Comput. Syst. Sci. Eng.* **2019**, *34*, 4.
6. Jehangiri, A.I.; Maqsood, T.; Ahmad, Z.; Umar, A.I.; Shuja, J.; Alanazi, E.; Alasmary, W. Mobility-aware computational offloading in mobile edge networks: A survey. *Clust. Comput.* **2021**, *24*, 2735–2756.
7. Yan, C.; Zhang, Y.; Zhong, W.; Zhang, C.; Xin, B. A truncated SVD-based ARIMA model for multiple QoS prediction in mobile edge computing. *Tsinghua Sci. Technol.* **2021**, *27*, 315–324. [CrossRef]
8. Li, L.; Zhao, G.; Blum, R.S. A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1710–1732. [CrossRef]
9. Safavat, S.; Sapavath, N.N.; Rawat, D.B. Recent advances in mobile edge computing and content caching. *Digit. Commun. Netw.* **2020**, *6*, 189–194. [CrossRef]
10. Singh, S.; Jeong, Y.-S.; Park, J.H. A survey on cloud computing security: Issues, threats, and solutions. *J. Netw. Comput. Appl.* **2016**, *75*, 200–222. [CrossRef]
11. Uzaman, S.K.; Shuja, J.; Maqsood, T.; Rehman, F.; Mustafa, S. A systems overview of commercial data centers: Initial energy and cost analysis. *Int. J. Inf. Technol. Web Eng. IJITWE* **2019**, *14*, 42–65. [CrossRef]
12. Ko, H.; Kyung, Y. Performance Analysis and Optimization of Delayed Offloading System With Opportunistic Fog Node. *IEEE Trans. Veh. Technol.* **2022**, *71*, 10203–10208. [CrossRef]
13. Mehrabi, M.; Salah, H.; Fitzek, F.H.P. A Survey on Mobility Management for MEC-enabled Systems. In *Proceedings of the 2019 IEEE 2nd 5G World Forum (5GWF), Dresden, Germany, 30 September 2019–2 October 2019*; IEEE: New York, NY, USA, 2019; pp. 259–263.
14. Zaman, S.K.U.; Jehangiri, A.I.; Maqsood, T.; Haq, N.U.; Umar, A.I.; Shuja, J.; Ahmad, Z.; Ben Dhaou, I.; Alsharekh, M.F. LiMPO: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing. *Clust. Comput.* **2022**, 1–19. [CrossRef]
15. Yasir, M.; Zaman, S.K.U.; Maqsood, T.; Rehman, F.; Mustafa, S. CoPUP: Content popularity and user preferences aware content caching framework in mobile edge computing. *Clust. Comput.* **2022**, 1–15. [CrossRef]
16. Zaman, S.K.U.; Khan, I.A.; Hussain, S.S.; Iqbal, T.; Shuja, J.; Ahmed, S.F.; Jararweh, Y.; Ko, K. PreDiKT-OnOff: A complex adaptive approach to study the impact of digital social networks on Pakistani students' personal and social life. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5121. [CrossRef]
17. Li, C.; Song, M.; Yu, C.; Luo, Y. Mobility and marginal gain based content caching and placement for cooperative edge-cloud computing. *Inf. Sci.* **2021**, *548*, 153–176. [CrossRef]
18. Keshavarzian, I.; Zeinalpour-Yazdi, Z.; Tadaion, A. Energy-efficient mobility-aware caching algorithms for clustered small cells in ultra-dense networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6833–6846. [CrossRef]
19. Somesula, M.K.; Rout, R.R.; Somayajulu, D. Contact duration-aware cooperative cache placement using genetic algorithm for mobile edge networks. *Comput. Netw.* **2021**, *193*, 108062. [CrossRef]
20. Li, H.; Sun, C.; Li, X.; Xiong, Q.; Wen, J.; Wang, X.; Leung, V.C.M. Mobility-Aware Content Caching and User Association for Ultra-Dense Mobile Edge Computing Networks. In *Proceedings of the GLOBECOM 2020-2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020*; IEEE: New York, NY, USA, 2020; pp. 1–6.
21. Huynh, L.N.T.; Pham, Q.-V.; Nguyen, T.D.T.; Hossain, D.; Shin, Y.-R.; Huh, E.-N. Joint Computational Offloading and Data-Content Caching in NOMA-MEC Networks. *IEEE Access* **2021**, *9*, 12943–12954. [CrossRef]
22. Liu, W.; Jiang, Y.; Xu, S.; Cao, G.; Du, W.; Cheng, Y. Mobility-aware video prefetch caching and replacement strategies in mobile-edge computing networks. In *Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), Singapore, 11–13 December 2018*; IEEE: New York, NY, USA, 2018; pp. 687–694.
23. AlNagar, Y.; Hosny, S.; El-Sherif, A.A. Towards mobility-aware proactive caching for vehicular ad hoc networks. In *Proceedings of the 2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW), Marrakech, Morocco, 15–18 April 2019*; IEEE: New York, NY, USA, 2019; pp. 1–6.
24. Bi, R.; Liu, Q.; Ren, J.; Tan, G. Utility aware offloading for mobile-edge computing. *Tsinghua Sci. Technol.* **2020**, *26*, 239–250. [CrossRef]
25. Li, C.; Zhang, Y.; Song, M.; Yan, X.; Luo, Y. An optimized content caching strategy for video stream in edge-cloud environment. *J. Netw. Comput. Appl.* **2021**, *191*, 103158. [CrossRef]
26. Yu, Z.; Hu, J.; Min, G.; Zhao, Z.; Miao, W.; Hossain, M.S. Mobility-Aware Proactive Edge Caching for Connected Vehicles Using Federated Learning. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 5341–5351. [CrossRef]
27. Wei, H.; Luo, H.; Sun, Y. Mobility-Aware Service Caching in Mobile Edge Computing for Internet of Things. *Sensors* **2020**, *20*, 610. [CrossRef] [PubMed]
28. Jiang, W.; Feng, G.; Qin, S.; Liang, .-C. Learning-based cooperative content caching policy for mobile edge computing. In *Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019*; IEEE: New York, NY, USA, 2019; pp. 1–6.

29. Ye, Y.; Xiao, M.; Skoglund, M. Mobility-Aware Content Preference Learning in Decentralized Caching Networks. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *6*, 62–73. [CrossRef]
30. Zaman, S.K.U.; Khan, A.U.R.; Malik, S.U.R.; Khan, A.N.; Maqsood, T.; Madani, S.A. Formal verification and performance evaluation of task scheduling heuristics for makespan optimization and workflow distribution in large-scale computing systems. *Comput. Syst. Sci. Eng.* **2017**, *32*, 227–241.
31. Liu, X.; Sun, C.; Zhang, X. Context-aware caching with social behavior in MEC-enabled wireless cellular networks. In *Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 11–15 March 2019*; IEEE: New York, NY, USA, 2019; pp. 1004–1008.
32. Zeng, Y.; Xie, J.; Jiang, H.; Huang, G.; Yi, S.; Xiong, N.; Li, J. Smart caching based on user behavior for mobile edge computing. *Inf. Sci.* **2019**, *503*, 444–468. [CrossRef]
33. Nguyen, T.-V.; Dao, N.-N.; Noh, W.; Cho, S. User-Aware and Flexible Proactive Caching using LSTM and Ensemble Learning in IoT-MEC Networks. *IEEE Internet Things J.* **2021**, *in press*. [CrossRef]
34. Narayanan, A.; Verma, S.; Ramadan, E.; Babaie, P.; Zhang, Z.-L. Deepcache: A deep learning based framework for content caching. In Proceedings of the 2018 Workshop on Network Meets AI & ML, New York, NY, USA, 24 August 2018; pp. 48–53.
35. Herlocker, J.L.; Konstan, J.A.; Terveen, L.G.; Riedl, J.T. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* **2004**, *22*, 5–53. [CrossRef]
36. Zhou, P.; Gong, S.; Xu, Z.; Chen, L.; Xie, Y.; Jiang, C.; Ding, X. Trustworthy and Context-Aware Distributed Online Learning With Autoscaling for Content Caching in Collaborative Mobile Edge Computing. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 1032–1047. [CrossRef]
37. Nath, S.; Wu, J. Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems. *Intell. Converg. Netw.* **2020**, *1*, 181–198. [CrossRef]
38. Xue, F.; Cui, Q.; Fu, T.; Wang, J. *Further Advancements for E-UTRA Physical Layer Aspects (Release 9)*; European Telecommunications Standards Institute: Sophia Antipolis, France, 2010.
39. Wu, W.; Zhou, F.; Hu, R.Q.; Wang, B. Energy-Efficient Resource Allocation for Secure NOMA-Enabled Mobile Edge Computing Networks. *IEEE Trans. Commun.* **2019**, *68*, 493–505. [CrossRef]