Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Lightweight similarity checking for English literatures in mobile edge computing

Xiaomei Liu[1], Ailing Gao[1], Chengxiang Chen[2] and Mohammad Mahdi Moghimi[3*]

Abstract

With the advent of information age, mobile devices have become one of the major convenient equipment that aids people's daily office activities such as academic research, one of whose major tasks is to check the repetition rate or similarity among different English literatures. Traditional literature similarity checking solutions in cloud paradigm often call for intensive computational cost and long waiting time. To tackle this issue, in this paper, we modify the traditional literature similarity checking solution in cloud paradigm to make it suitable for the light-weight mobile edge environment. Furthermore, we put forward a lightweight similarity checking approach SC_{MEC} for English literatures in mobile edge computing environment. To validate the advantages of SC_{MEC}, we have designed massive experiments on a dataset. The reported experimental results show that SC_{MEC} can deliver a satisfactory similarity checking result of literatures compared to other existing approaches.

Keywords Mobile edge computing, Literature similarity checking, Hash index, Lightweight

Introduction

Benefiting from the continuous progress of mobile computing technology, various mobile devices (e.g., smart phones, PDA, etc.) have engaged into the daily life of people and are more and more crucial in people's entertainment, job, shopping and so on [1–3]. Such a mobile life manner brings many conveniences to people since it minimizes the negative influences brought by physical space of different people [4–7]. Specifically, with the advent of COVID-19 pandemic, people's activity ranges are further limited. In this situation, mobile devices have gradually become one of the major ways to aid people's various activities including academic research [8–10]. Today, more and more people are apt to take mobile devices as their major academic research tools and hence generate a series of mobile devices-enabled academic research tools such as online academic meeting, multi-party academic collaborations, etc. [11–13].

As one of the major tasks in academic activities, literature similarity checking is playing an increasingly important role in ensuring the success of scientific research. However, in the mobile computing environment, English literatures as well as their corresponding user-literature reading records should be sent to a remote cloud platform for uniform similarity checking (e.g., through a collaborative filtering manner) in a centralized manner, which raises a heavy burden on the response time of similarity checking in cloud platform since there are so many literatures as well as historical user-literature records in various academic databases. In addition, in the traditional cloud-based literature similarity calculation process, each user's literature reading records are a kind of sensitive information that calls for certain privacy protection. Therefore, it becomes particularly emergent to seek for other privacy-preserving and lightweight literature similarity checking solutions to accommodate the personalized requirements of researchers in the mobile computing environment. Fortunately, edge computing has shown its unique advantages in processing various big data

*Correspondence:
Mohammad Mahdi Moghimi
ms.moghimi@iauyazd.ac.ir
[1] Shandong Provincial University Laboratory for Protected Horticulture, Weifang University of Science and Technology, Weifang, China
[2] Fujian Polytechnic Normal University, Fuzhou, China
[3] Department of Electrical Engineering, Yazd Branch, Islamic Azad University, Tehran, Iran

mining and analysis tasks due to its inherent properties in low data transmission amount and quick response time [14–17]; because in edge computing paradigm, most data are not necessary sent to a cloud center for integration, instead, most data are pre-processed beforehand in the nearby edge servers. This way, the data transmission and response time conditions are improved considerably [18–21].

However, mobile edge computing is still on the early stage after its birth. And therefore, there are still some critical issues that need to be solved well [22–25]. Especially, how to perform lightweight similarity checking of English literatures in the mobile computing environment for easy-to-use mobile office services is a difficult task that needs to be further studied. In view of this challenge, a hash-based index mechanism is introduced in this paper (the time complexity of this hash technique has been proven to be close to O(1)), which is followed by a lightweight literature similarity checking approach based on literature indexes, i.e., SC_{MEC} is proposed to alleviate the heavy computational costs and transmission costs in mobile academic activities.

Generally, our contributions are three-fold.

(1) An index mechanism is proposed in our paper to aid the literature similarity checking task in the mobile edge computing, in which most literature data from mobile devices are quickly processed by the closer edge servers, without being sent to the remote cloud platform for uniform but time-consuming processed.

(2) An index-based literature similarity checking approach SC_{MEC} is proposed to support the academic activities on mobile devices. Since the index mechanism is proven to be of a low time complexity, our proposed index-based literature similarity checking approach SC_{MEC} is very lightweight and time-efficient.

(3) Experiments are deployed for validation and the final experimental results with related approaches prove the effectiveness and efficiency of our SC_{MEC} in this paper.

Our paper is structured as follows. Related literatures are investigated in Section 2. A three-layer framework is presented in Section 3 to support the literature similarity checking tasks in the mobile edge computing environment. An index-based literature similarity checking approach SC_{MEC} is put forward in Section 4. Evaluations are presented in Section 5. At last, we summarize the paper in Section 6 which also points out the future research directions.

Related literature

The literature similarity checking issue in cloud environment has been investigated by researchers. Next, we summarize existing literatures as follows.

In [26], with the advent of cloud computing, data management jobs are outsourced to cloud to save money. However, there are privacy issues. This paper considers the use of cloud to encrypt data before data outsourcing, and conducts similarity search of multiple keywords on outsourced cloud data. The experiment proves that the search design implemented in this paper can effectively resist internal threats and show high performance in cloud search time. In [27], to solve security problem of cloud, the authors bring forth a verifiable privacy-protected multi-keyword text search (MTS) solution based on similarity ranking. Meanwhile, to improve search efficiency of data, this paper also suggested a tree-based index structure as well as a variety of adaptive multidimensional (MD) algorithms, which make the actual search efficiency far better than the linear search efficiency. In [28], aiming at the protection of cloud data privacy, this paper proposed a new similarity based secure data deduplication scheme combining bloom filter and content definition chunking technology. The scheme only deduplicates similar files to significantly reduce computing overhead. In [29], deduplication is widely used in cloud computing to improve space complexity. While the secure mechanisms have some security drawbacks, such as the inability to provide flexible access control. In this paper, the authors propose an encrypted deduplication scheme EDedup based on similarity awareness, which supports flexible revocable access control. EDedup divides the file into segments, which take advantage of similarity to reduce computation overhead through a representative hash algorithm. EDedup also integrates source-based similarity detection and target-based duplicate block detection to protect against attacks and ensure efficient deduplication.

In [30], considering the limitations of classical ICP approaches, this paper proposes a improved iterative nearest point (ICP) approach by the similarity of point cloud curvature features. Based on the classical ICP algorithm, the authors introduce the rough alignment method of principal component analysis, and use k-D tree to segment three-dimensional point cloud to fasten the search process of nearest neighbor points. Experiments show that the approach is more accurate. In [31], more recently, the emergence of new applications using advanced content representation has driven the rise of immersion technology. Among the alternatives available, the point cloud is a promising solution. Here, an effective objective measure is introduced to capture the perceptual degradation of distorted point clouds. The quality assessment of point clouds is mainly based on angular similarity. In [32], the QoS of cloud services will change gradually over time, but the existing service recommendation methods do not pay attention to this problem. Therefore, the authors bring forth a time-aware recommendation method to solve this problem. A new similarity enhanced collaborative filtering method is developed to capture the temporal characteristics of user similarity and solve the problem of data sparsity. Meanwhile, ARIMA model is used for service quality prediction at future time points. In [33], in addition to cloud data storage services, data similarity retrieval is another basic service provided by cloud, especially in image data. This paper

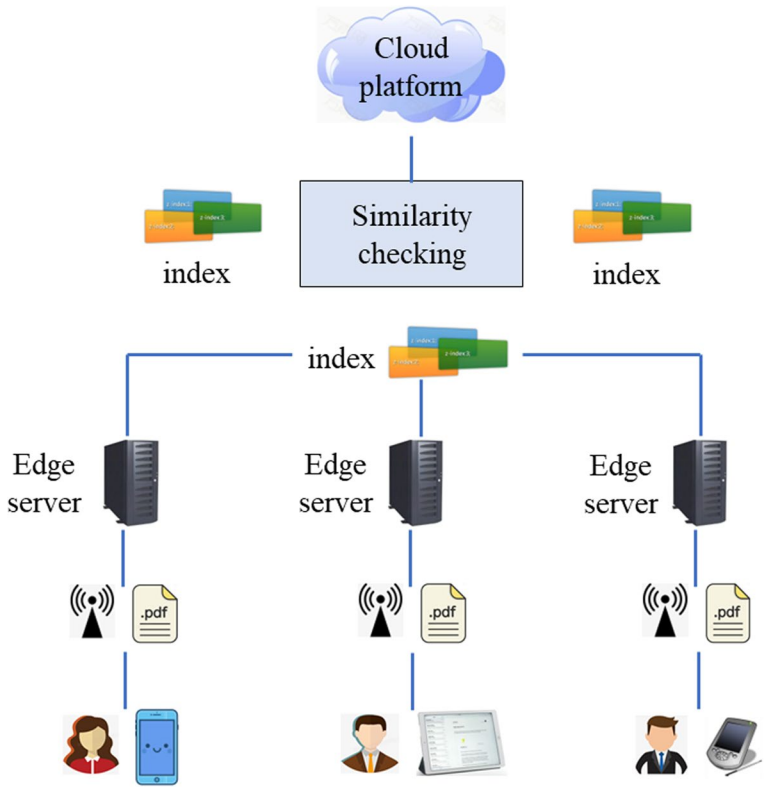

Fig. 1 Three-layer framework of literature similarity checking in mobile edge computing

proposes a privacy-protecting similar image detection solution based on LSH for image hosted in cloud. This method can effectively secure the sensitive cloud data and obtain better retrieval results.

With the above investigation, a simple conclusion could be drawn that existing literatures are more focusing on the delay, security and privacy of cloud-based literature similarity checking, without considering the migration of literature similarity checking task from cloud to mobile edge, which limits the success of mobile academic activities considerably. Considering this limitation, we put forward an index-based literature similarity checking solution SC_{MEC} in the next sections.

Framework

A three-layer framework is presented in Fig. 1 to support the literature similarity checking tasks in the mobile edge computing environment. As Fig. 1 shows, researchers use various mobile devices (e.g., mobile phone, iPAD, PDA and so on [34–36]) for different academic activities. During this process, the academic literatures on mobile devices [37, 38] (e.g., PDFs) are not necessary to be transmitted directly to the remote cloud platform. Instead, researchers transmit their academic literatures to nearby edge servers through close mobile devices via wireless communication technology. Afterwards, each edge server is responsible for converting the academic literatures hosted in the edge server into a corresponding index (we call it literature index here). The literature index here is a lightweight embedding of the literature compared to the literature itself. Each edge server records its converted literature indexes and transmits them to a cloud platform that is responsible for processing all the literature indexes transmitted from all edge servers. Finally, the cloud platform checks the literature similarity via comparing the literature indexes which are very short and lightweight. This way, we can achieve the literature similarity checking task through the three-layer device-edge-cloud framework presented in Fig. 1.

Index-based similarity checking in mobile edge computing: SC_{MEC}

According to the three-layer framework of literature similarity checking task presented in Fig. 1, our proposed literature similarity checking approach SC_{MEC} can be divided into two steps: first, we need to convert each literature from mobile devices into its corresponding index; second, we need to send the literature indexes to the central cloud platform for uniform similarity checking. Next, we introduce the major two steps of SC_{MEC} approach.

Step 1: Conversion from literatures to indexes.

Traditional literature similarity checking approaches need to make comparisons between different literatures

in a direct and straightforward way, e.g., word by word, sentence by sentence, paragraph by paragraph, etc. Such a direct and straightforward literature comparison way is often time-consuming and tiresome [39, 40], which probably decreases the satisfaction degree of researchers who often expect a quick and accurate literature similarity checking result. Therefore, to speed up the above literature similarity checking process, we need to convert the initial literatures which are often long into corresponding shorter embeddings. Since the content of each literature is often much, we convert each literature into a short index or embedding. This way, we can evaluate whether two literatures are similar or not by comparing their index values instead of their literature contents. The advantage of such an operation is that we can minimize the time cost for similar literature evaluation and discovery. Here, we adopt the classic Simhash and LSH techniques (the time complexity of the mentioned Simhash and LSH techniques have been proven to be close to O(1)). In our proposal, we use Simhash technique to convert each literature into a corresponding Boolean vector, whose purpose is to convert the text information of each literature into a corresponding 0/1 string that is easy to process and calculate in the subsequent similar literature evaluation process.

Next, we introduce the concrete step of conversion from literatures to indexes. First, we use Simhash technique to convert each literature (here, the literature set is denoted by $Lit_{Set} = (lit_1, \ldots, lit_n)$) into a long signature (here, the signature set for the literatures in Lit_{Set} is denoted by $Sig_{Set} = (sig_1, \ldots, sig_n)$). The concrete conversion process is introduced in detail at follows.

(1) Word segmentation for literatures.

Each literature often includes many words, which makes it hard to calculate the similarity degree between different literatures directly. To tackle this issue, we first convert a long literature into a word vector through various mature word segmentation tools in natural language processing (NLP) domain, e.g., word2vec or fastText. The concrete word segmentation process will not be introduced in detail here. Interested readers can refer to the related literatures in NLP. Here, we take a literature *lit* as an example for illustration. We assume that the literature *lit* is converted into a word vector V_{lit} as specified in (1) based on word2vec or fastText. Here, m is not a fixed value since different literatures often include different word number after word segmentation.

$$V_{lit} = (a_1, \ldots, a_m) \tag{1}$$

(2) Hash projection from a word vector to a 0/1 vector.

In the last substep, we have converted each literature *lit* into a word vector V_{lit}. However, it is challenging to evaluate the similarity between word vectors. To tackle this issue, we further convert the word vector V_{lit} of literature *lit* into a numerical vector $NumV_{lit}$. The concrete conversion process is based on any hash projection table. Here, for simplicity, we use the classic ASCII coding table adopted widely in computer domain to achieve the goal of hash projection. For example, if $V_{lit} = (a_1, a_2)$, $a_1 = 11110000$ and $a_2 = 10101010$, then $NumV_{lit} = (1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0)$. Next, we replace the "0" entries in the vector by "-1" and then derive a new vector constituted by "1" and "-1" entries only. For example, $NumV_{lit} = (1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0)$ is updated to be $NumV_{lit} = (1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ 1\ -1\ 1\ -1\ 1\ -1)$.

(3) Vector weighting.

Weight significance is often inevitable in many applications involving multiple dimensions or criteria [41–43]. Inspired by the above analysis, in a literature, each word should be assigned a concrete weight value indicating the importance and significance of the word in depicting the whole literature. The weights of words in a literature could be generated in many ways such as TF/IDF, which is not repeated here. Here, we assume that weight vector corresponding to the m words in vector V_{lit} in (1) is W as specified in (2)-(3).

$$W_{lit} = (w_1, \ldots, w_m) \tag{2}$$

$$\sum_{j=1}^{m} w_j = 1 \tag{3}$$

(4) Vector union by addition.

For a literature *lit*, with its word vector $V_{lit} = (a_1, \ldots, a_m)$ in Eq. (1) (here, please note that the hash projection from 0 to -1 has been applied to the word vector) and its weight vector $W_{lit} = (w_1, \ldots, w_m)$ in Eq. (2), we can make a dot production operation between vectors V_{lit} and W_{lit}, whose result is denoted by DP_{lit} in (4). For example, if $V_{lit} = (a_1, a_2)$, $a_1 = 11110000$, $a_2 = 10101010$, $w_1 = 0.4$, $w_2 = 0.6$, then $DP_{lit} = (1\ 1\ 1\ 1\ -1\ -1\ -1\ -1)$ * $0.4 + (1\ -1\ 1\ -1\ 1\ -1\ 1\ -1)$ * $0.6 = (0.4\ 0.4\ 0.4\ 0.4\ -0.4\ -0.4\ -0.4\ -0.4) + (0.6\ -0.6\ 0.6\ -0.6\ 0.6\ -0.6\ 0.6\ -0.6) = (1\ -0.2\ 1\ -0.2\ 1\ -0.2\ 1\ -0.2)$.

$$DP_{lit} = (b_1, \ldots, b_m) = V_{lit} * W_{lit} = \sum_{j=1}^{m} a_j * w_j \tag{4}$$

(5) Dimension reduction.

In the above substep, we have converted each literature *lit* into a corresponding $DP_{lit} = (b_1, \ldots, b_m)$ by Eq. (4). However, from the dimension perspective, each entry in vector DP_{lit} can take any real value and therefore, its value range is often very large and not suitable for subsequent similarity calculation and evaluation. To overcome this shortcoming, we reduce each dimension's value range as follows, since binary embedding is widely applied to various big data scenarios to reduce the search and processing time [44–46]. In concrete, we make the following conversions in Eq. (5). Afterwards,

each entry b_j in vector DP_{lit} is equal to either 1 or 0, which narrows the value range of vector DP_{lit} significantly. This way, we successfully achieve the goal of dimension reduction.

$$b_j = \begin{cases} 1 \text{ if } b_j > 0, \\ 0 \text{ if } b_j \leq 0. \end{cases} (j = 1, 2, \ldots, m) \quad (5)$$

(6) Deep dimension reduction by LSH.

To further reduce the dimensions of DP_{lit} for each literature *lit*, we use LSH technique to build a deep index for each *lit*. Concretely, we generate an m-dimensional vector $X = (x_1, \ldots, x_m)$ randomly by Eq. (6), where each entry x_j of vector X belongs to range [-1, 1]. Next, we convert the m-dimensional vector DP_{lit} into a smaller r-dimensional vector$Z (r \ll m)$ by Eqs. (7)-(9). Here, the purpose of Eq. (7) is to calculate the projection from the original vector DP_{lit} to the vector X; the purpose of Eq. (8) is to reduce the dimensions involved; afterwards, we repeat the operations in Eqs. (7) and (8) r times to obtain $z_1, \ldots, z_r$. Then we can get a new vector $Z = (z_1, \ldots, z_r)$ which is much shorter than the original vector DP_{lit}. This way, we successfully achieve the goal of dimension reduction. In addition, the LSH technique has been proven a lightweight nearest neighbor discovery approach whose time complexity is approximately O(1). Therefore, the proposed LSH-based similar literature discovery approach is very suitable for the big data context.

$$x_j = random(-1, 1) \quad (6)$$

$$z = DP_{lit} * X = \sum_{j=1}^{m} b_j * x_j \quad (7)$$

$$z = \begin{cases} 1 \text{ if } z > 0, \\ 0 \text{ if } z \leq 0. \end{cases} \quad (8)$$

$$Z = (z_1, \ldots, z_r) \quad (9)$$

Step 2: Similarity checking of literatures based on indexes.

In Step 1, for each literature in $Lit_{Set} = (lit_1, \ldots, lit_n)$, we have obtained a corresponding hash index $Z_j (j = 1, 2, \ldots, n)$. Next, we compare any two literatures lit_i and lit_j $(1 \leq i \leq n, 1 \leq j \leq n)$ through comparing their respective hash indexes Z_i and Z_j. In concrete, if $Z_i = Z_j$ holds, we can simply conclude that literatures lit_i and lit_j are similar with high probability. However, the above literature similarity evaluation solution is not always correct since LSH is a probability-based neighbor search technique. To minimize the negative influences incurred by probability, for each literature lit_j in Lit_{Set}, we do not generate only one hash index Z_j; instead, we generate h indexes $Z_j^1, \ldots, Z_j^h$. Afterwards, the similarity between literatures lit_i and lit_j is evaluated by Eq. (10). This way, we can evaluate whether two literatures are similar or same based on their respective hash indexes, to improve the literature similarity evaluation efficiency.

$$lit_i \text{ and } lit_j \text{ are similar iff } \exists\ k \text{ satisfying } Z_i^k = Z_j^k (1 \leq k \leq h) \quad (10)$$

The details of our proposed SC_{MEC} algorithm can be described clearly by the following pseudo code.

Require: (1)$Lit_{Set} = (lit_1, \ldots, lit_n)$: n literatures
(2) $W_{lit} = (w_1, \ldots, w_m)$: word weight vector
(3) r: number of hash functions
(4) h: number of hash tables
Ensure: (1) $TB = (tb_1, \ldots, tb_y)$: similar literature set

for each $lit \in Lit_{Set}$ **do**
  $V_{lit} = (a_1, \ldots, a_m)$ by word2vec or fastText
  Convert V_{lit} into $NumV_{lit}$ by ASCII table
  Replace "0" in $NumV_{lit}$ by "-1"
  Update $V_{lit} = NumV_{lit}$
  Calculate $DP_{lit} = V_{lit} * W_{lit}$ by Eq.(1)
  $(b_1, \ldots, b_m) = DP_{lit}$
  for each $b_j \in b_1, \ldots, b_m$ **do**
    if then$b_j > 0$
      $b_j = 1$
    else
      $b_j = 0$
    end if
  end for
end for
for each $x_j \in x_1, \ldots, x_m$ **do**
  $x_j =$ random (-1, 1)
end for
$X = (x_1, \ldots, x_m)$
for each $lit \in Lit_{Set}$ **do**
  Calculate $z = DP_{lit} * X$ by Eq.(7)
  if then$z > 0$
    $z = 1$
  else
    $z = 0$
  end if
end for
for $p = 1$ to r **do**
  repeat lines $1 \sim 25$
end for
$Z = (z_1, \ldots, z_r)$
for $k = 1$ to h **do**
  repeat lines $26 \sim 30$ to generate $Z_1, \ldots, Z_h$
end for
for any lit_i , $lit_j \in Lit_{Set}$ **do**
  if there exists $k \in \{1, \ldots, h\}$ satisfying $Z_i^k = Z_j^k$ **then**
    lit_i and lit_j are similar
    Put lit_i and lit_j into tb
  end if
end for
return $TB = (tb_1, \ldots, tb_y)$

Algorithm 1 SC_{MEC}

Evaluation

Next, we prove the feasibility of SC_{MEC} algorithm in handling literature similarity checking in mobile edge computing environment. Concretely, the experiments

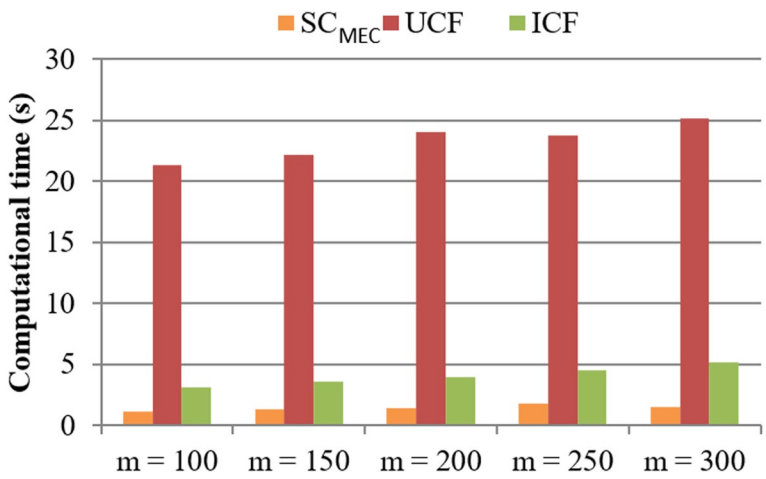

Fig. 2 Computational time comparison (w.r.t. *m*)

are deployed on WS-DREAM dataset[1] which records a group of services as well as their quality performances. Here, we use the records in the dataset to simulate the literatures as well as their digital signatures. Two classic similar neighbor searching methods are used for comparison purpose, i.e., UCF and ICF. Experiment hardware includes a 2.50 GHz processor and 16.0 GB memory; experiments are executed under WIN-7 operation system and Python-3. We run each experiment 100 times and record their average performance for result display. Please note that the parameter settings in this paper are determined in an experienced way, i.e., by experiment tests.

Test 1: Computational time comparison.

In the big data context, data processing speed is very important since response time is an influencing factor associated with user experience [47, 48]. Inspired by this observation, we test the speed of SC_{MEC} with baseline methods: UCF and ICF. Test results are presented in Figs. 2 and 3, respectively. In concrete, in Fig. 2, time cost of three methods is compared with the size of signature dimensions, i.e., m. Here, m varies from 100 to 300; n is equal to 5000; number of functions in SC_{MEC}, i.e., $r = 10$; number of tables in SC_{MEC}, i.e., $k = 18$. In Fig. 2, the time costs of three methods approximate rise with the rising of m since more calculation operations are necessary in all three methods when there are more signature dimensions of literatures. Moreover, compared to UCF and ICF methods, our proposed SC_{MEC} algorithm consumes less time. This is due to the fact that UCF and ICF need to calculate literature similarity based on collaborative filtering idea whose time complexity is relatively high; while the SC_{MEC} algorithm first generates literature indexes offline and then uses existing literature indexes to discover similar literatures. Therefore, SC_{MEC} often performs better than UCF and ICF in terms of consumed time. Similar comparison results could be observed in Fig. 3 where the computational time of three methods is compared with the number of literatures, i.e., n. Here, n varies from 1000 to 5000, m is equal to 300, $r = 10$, $k = 18$. The reason is not repeated again.

Test 2: Accuracy comparison. Here, we measure the literature similarity checking accuracy (in the form of MAE, smaller is better) of three methods, whose results are presented in Figs. 4 and 5, respectively. In concrete, in Fig. 4, $m = \{100, \ldots, 300\}$, $n = 5000$, $r = 10$, $k = 18$. The results reported in Fig. 4 shows that the accuracy of three methods all varies with the growth of m and SC_{MEC} algorithm outperforms UCF and ICF methods in accuracy. The reason is that in SC_{MEC}, the adopted hash indexes of literatures can guarantee to discover all the most similar literatures. As a result, the accuracy of SC_{MEC} is the highest and the MAE of SC_{MEC} is the lowest. Such a comparison result indicates a good literature similarity checking performance of the proposed SC_{MEC} method. Similar results could also be observed in Fig. 5 where $n = \{1000, \ldots, 5000\}$, $m = 300$, $r = 10$, $k = 18$. Figure 5 shows that the accuracy of three methods approximately rises when n (i.e., MAE of three methods declines when n rises) grows. The reason can be analyzed as follows: when n becomes larger, there are more literatures as well as their associated signatures; in this situation, more valuable information taking part in literature similarity checking is available and hence, the accuracy is enhanced accordingly. Furthermore, the MAE of SC_{MEC} is generally lower than UCF and ICF, which indicates a better performance of SC_{MEC} in literature similarity checking.

Test 3: Performances of SC_{MEC}.

[1] https://wsdream.github.io/

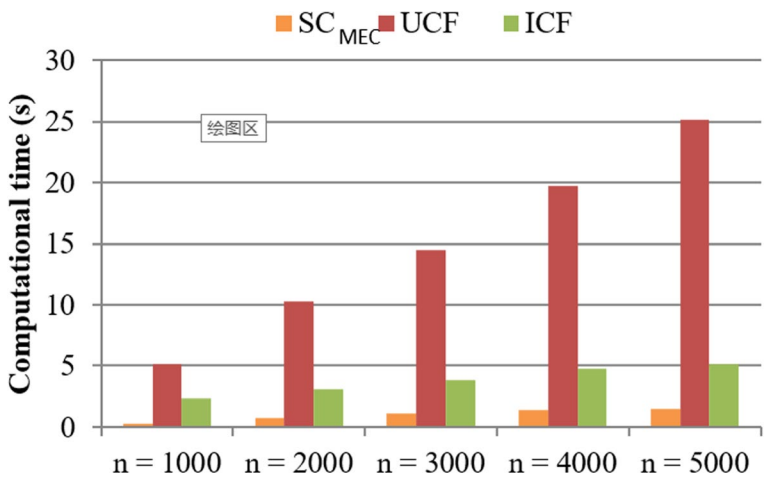

Fig. 3 Computational time comparison (w.r.t. *n*)

As introduced in Section 4, there are several key influencing factors that can determine the performances of SC_{MEC}, such as parameters r and k. To observe the relationships between the influencing factors and the algorithm performances, several experiments are designed where n and m are equal to 5000 and 300, respectively; r varies from 2 to 10 and k varies from 10 to 18. Concrete data are reported in Figs. 6 and 7. In concrete, Fig. 6 shows the computational time of SC_{MEC} when the parameters r and k fluctuate, where the computational time approximately drops with the increment of r and the consumed time approximately drops with the increment of parameter k. Next, we discuss the reason behind such a conclusion. First, if we use more hash functions (i.e., larger r) used to calculate literature similarity, the more probable that the top-similar literatures be discovered; in this situation, the finally returned similar literatures are usually fewer and hence, less time is consumed accordingly. Second, if we use more hash tables (i.e., larger k) used to calculate literature similarity, more similar literatures are probably discovered; in this situation, the finally returned similar literatures are usually more and hence, more time is consumed accordingly.

Figure 7 shows the accuracy of SC_{MEC} when the parameters r and k fluctuate, where the accuracy value approximately drops with the increment of parameter r and the accuracy varies with the increment of parameter k. Next, like in Fig. 6, we discuss the reason behind such an observation from Fig. 7. If we use more hash functions (i.e., larger r) to calculate literature similarity, the returned

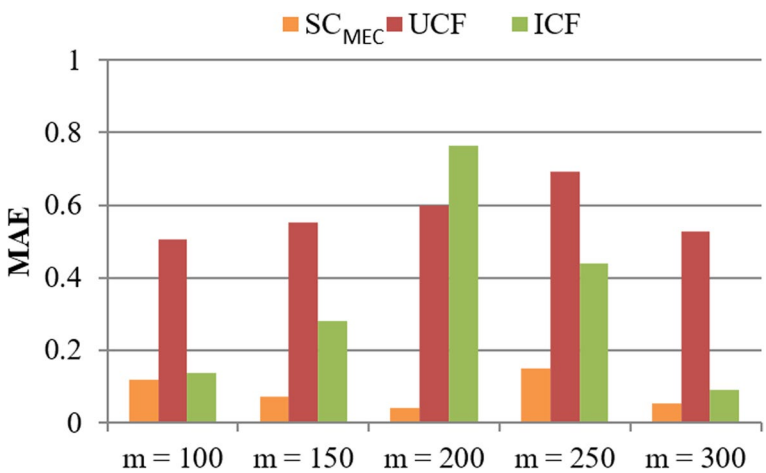

Fig. 4 Accuracy comparison (w.r.t. *m*)

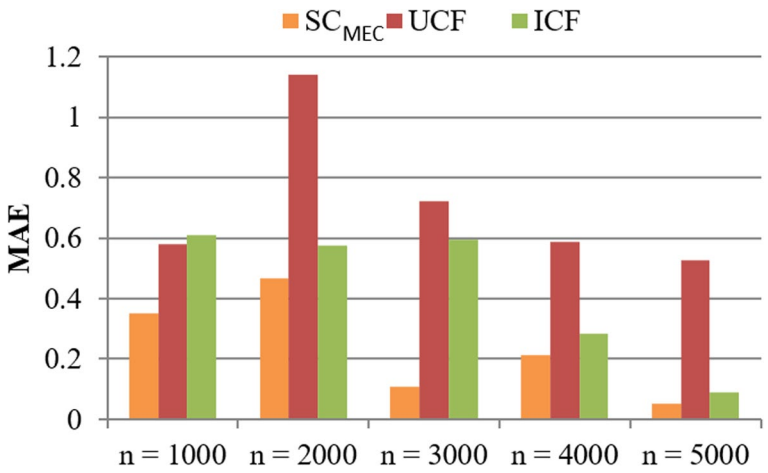

Fig. 5 Accuracy comparison (w.r.t. *n*)

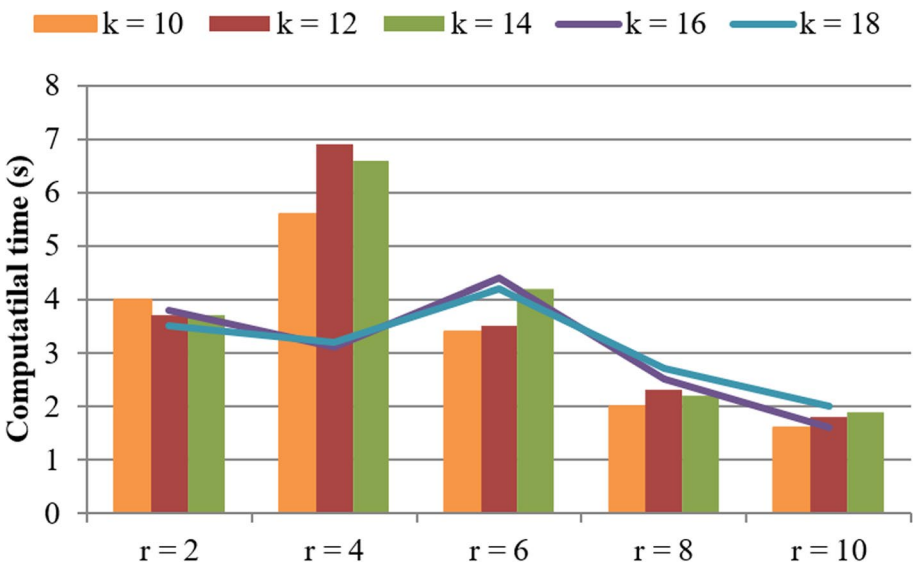

Fig. 6 Computational time of our algorithm (w.r.t. *r* and *k*)

similar literatures are "more similar"; accordingly, literature similarity checking accuracy is higher (i.e., MAE is lower).

Conslusions

With the advent of information age, mobile devices have become one of the major convenient equipment that aids people's daily office activities such as academic research, one of whose major tasks is to check the repetition rate or similarity among different English literatures. Traditional literature similarity checking solutions in cloud paradigm often call for intensive computational cost and long waiting time. To tackle this issue, in this paper, we modify the traditional literature similarity checking solution in cloud paradigm to make it more suitable for the light-weight mobile edge environment. Furthermore, we put forward a light-weight similarity checking approach SC_{MEC} for English literatures in mobile edge computing environment. To validate the advantages of SC_{MEC}, we have designed massive experiments on a dataset. The reported experimental results show that SC_{MEC} can deliver a satisfactory similarity checking result of literatures compared to other existing approaches.

Literature similarity checking in mobile edge computing environment often involves certain sensitive user

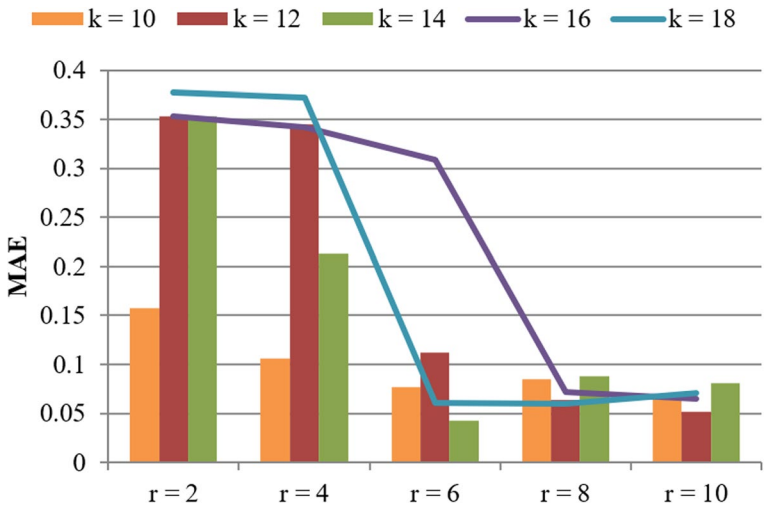

Fig. 7 Accuracy of our algorithm (w.r.t. *r* and *k*)

privacy [49]. Therefore, in future study, we will further consider the privacy disclosure issue in our proposed algorithm by introducing more effective privacy protection techniques including blockchain, federated learning, DP, encryption, etc. [50–52]. In addition, energy saving and cost optimization are also key challenges in typical big data applications [53, 54]. Hence, we will introduce more classic performance optimization and computational offloading technologies.

Abbreviations

MTS	Multi-keyword text search
ICP	Iterative nearest point
MD	Adaptive multidimensionaL
LSH	Locality-Sensitive Hashing
NLP	Natural language processing
MEC	Mobile edge computing

Acknowledgements
None.

Authors' contributions
Xiaomei Liu: idea, algorithm design; Ailing Gao: background and motivation; Chengxiang Chen: model design; Mohammad Mahdi Moghimi: English writing, experiments. The author(s) read and approved the final manuscript.

Funding
None.

Availability of data and materials
http://wsdream.github.io/

Declarations

Ethics approval and consent to participate
None.

Consent for publication
Not applicable.

Competing interests
The authors declare that there is no competing interests.

Received: 28 October 2022 Accepted: 23 December 2022
Published online: 05 January 2023

References

1. Li Y, Liu J, Cao B, Wang C (2018) Joint optimization of radio and virtual machine resources with uncertain user demands in mobile cloud computing. IEEE Trans Multimed 20(9):2427–2438
2. Xie Y, Gui FX, Wang WJ, Chien CF (2021) A two-stage multi-population genetic algorithm with heuristics for workflow scheduling in heterogeneous distributed computing environments. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC20213137881
3. Zhou D, Xue X, Zhou Z (2022) SLE2: the improved social learning evolution model of cloud manufacturing service ecosystem. IEEE Trans Ind Inform. https://doi.org/10.1109/TII.2022.3173053
4. Liu Y, Song Z, Xu X, Rafique W, Zhang X, Shen J, Khosravi MR, Qi L (2022) Bidirectional gru networks-based next poi category prediction for healthcare. Int J Intell Syst 37(7):4020–4040
5. Xu Y, Liu Z, Zhang C, Ren J, Zhang Y, Shen X (2021) Blockchain-based trustworthy energy dispatching approach for high renewable energy penetrated power systems. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2021.3117924
6. Chen Y, Xing H, Ma Z, et al (2022) Cost-efficient edge caching for noma-enabled IoT services. China Commun
7. Zhou J, Cao K, Zhou X, Chen M, Wei T, Hu S (2021) Throughput-conscious energy allocation and reliability-aware task assignment for renewable powered in-situ server systems. IEEE Trans Comput Aided Des Integr Circ Syst 41(3):516–529
8. Zhou X, Liang W, Li W, Yan K, Shimizu S, Wang K (2021) Hierarchical adversarial attacks against graph neural network based IoT network intrusion detection system. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2021.3130434
9. Chen Y, Zhao J, Wu Y et al (2022) Qoe-aware decentralized task offloading and resource allocation for end-edge-cloud systems: A game-theoretical approach. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2022.3223119
10. Yang W, Li X, Wang P, Hou J, Li Q, Zhang N (2022) Defect knowledge graph construction and application in multi-cloud IoT. J Cloud Comput 11(1):1–12

11. Dai H, Wang X, Lin X, Gu R, Shi S, Liu Y et al (2021) Placing wireless chargers with limited mobility. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2021.3136967
12. Qi L, Lin W, Zhang X, Dou W, Xu X, Chen J (2022) A correlation graph based approach for personalized and compatible web apis recommendation in mobile app development. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE20223168611
13. Tsai S-B (2021) Organization and user computing and risk control under social network preface. IGI GLOBAL, Hershey
14. Dai H, Yu J, Li M, Wang W, Liu AX, Ma J, Qi L, Chen G (2022) Bloom filter with noisy coding framework for multi-set membership testing. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE20223199646
15. Chen Y, Zhao F, Chen X, Wu Y (2022) Efficient multi-vehicle task offloading for mobile edge computing in 6g networks. IEEE Trans Veh Technol 71(5):4584–4595. https://doi.org/10.1109/TVT.2021.3133586
16. Kong L, Wang L, Gong W, Yan C, Duan Y, Qi L (2022) Lsh-aware multitype health data prediction with privacy preservation in edge environment. World Wide Web 25(5):1793–1808
17. Zhang C, Xu Y, Hu Y, Wu J, Ren J, Zhang Y (2021) A blockchain-based multi-cloud storage data auditing scheme to locate faults. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2021.3057771
18. Gu R, Chen Y, Liu S, Dai H, Chen G, Zhang K, Che Y, Huang Y (2021) Liquid: Intelligent resource estimation and network-efficient scheduling for deep learning jobs on distributed gpu clusters. IEEE Trans Parallel Distrib Syst 33(11):2808–2820
19. Zhou X, Xu X, Liang W, Zeng Z, Yan Z (2021) Deep-learning-enhanced multitarget detection for end–edge–cloud surveillance in smart IoT. IEEE Internet Things J 8(16):12588–12596
20. Xu J, Li D, Gu W et al (2022) Uav-assisted task offloading for IoT in smart buildings and environment via deep reinforcement learning. Build Environ 222. https://doi.org/10.1016/j.buildenv.2022.109218
21. Chen Y, Zhao F, Lu Y, Chen X (2023) Dynamic task offloading for mobile edge computing with hybrid energy supply. Tsinghua Sci Technol. https://doi.org/10.26599/TST.2021.9010050
22. Gu R, Zhang K, Xu Z, Che Y, Fan B, Hou H, Dai H, Yi L, Ding Y, Chen G, et al (2022) Fluid: Dataset abstraction and elastic acceleration for cloud-native deep learning training jobs. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE). IEEE, pp 2182–2195
23. Li Y, Xia S, Zheng M, Cao B, Liu Q (2022) Lyapunov optimization based trade-off policy for mobile cloud offloading in heterogeneous wireless networks. IEEE Trans Cloud Comput 10(1):491–505
24. Qi L, Yang Y, Zhou X, Rafique W, Ma J (2022) Fast anomaly identification based on multi-aspect data streams for intelligent intrusion detection toward secure industry 4.0. IEEE Trans Ind Inform 18(9):6503–6511
25. Wang F, Li G, Wang Y, Rafique W, Khosravi MR, Liu G, Liu Y, Qi L (2022) Privacy-aware traffic flow prediction based on multi-party sensor data with zero trust in smart city. ACM Trans Internet Technol. https://doi.org/10.1145/3511904
26. Yu CM, Chen CY, Chao HC (2015) Privacy-preserving multikeyword similarity search over outsourced cloud data. IEEE Syst J 11(2):385–394
27. Sun W, Wang B, Cao N, Li M, Lou W, Hou YT, Li H (2013) Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Trans Parallel Distrib Syst 25(11):3025–3035
28. Liu J, Wang J, Tao X, Shen J (2017) Secure similarity-based cloud data deduplication in ubiquitous city. Pervasive Mob Comput 41:231–242
29. Zhou Y, Feng D, Hua Y, Xia W, Fu M, Huang F, Zhang Y (2018) A similarity-aware encrypted deduplication scheme with flexible access control in the cloud. Futur Gener Comput Syst 84:177–189
30. Yao Z, Zhao Q, Li X, Bi Q (2021) Point cloud registration algorithm based on curvature feature similarity. Measurement 177(109):274
31. Alexiou E, Ebrahimi T (2018) Point cloud quality assessment metric based on angular similarity. In: 2018 IEEE International Conference on Multimedia and Expo (ICME). IEEE, pp 1–6
32. Ding S, Li Y, Wu D, Zhang Y, Yang S (2018) Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and arima model. Decis Support Syst 107:103–115
33. Wu Y, Wang X, Jiang ZL, Li X, Li J, Yiu SM, Liu Z, Zhao H, Zhang C (2018) Towards secure cloud data similarity retrieval: Privacy preserving near-duplicate image data detection. In: International Conference on Algorithms and Architectures for Parallel Processing. Springer, pp 374–388
34. Wang F, Zhu H, Srivastava G, Li S, Khosravi MR, Qi L (2022) Robust collaborative filtering recommendation with user-item-trust records. IEEE Trans Comput Soc Syst 9(4):986–996
35. Zhou X, Li Y, Liang W (2020) Cnn-rnn based intelligent recommendation for online medical pre-diagnosis support. IEEE/ACM Trans Comput Biol Bioinforma 18(3):912–921
36. Li K, Zhao J, Hu J et al (2022) Dynamic energy efficient task offloading and resource allocation for noma-enabled IoT in smart buildings and environment. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109513
37. Huang J, Gao H, Wan S et al (2023) Aoi-aware energy control and computation offloading for industrial IoT. Futur Gener Comput Syst 139:29–37
38. Chen Y, Gu W, Xu J, et al (2022) Dynamic task offloading for digital twin-empowered mobile edge computing via deep reinforcement learning. China Commun
39. Xu Y, Zhang C, Wang G, Qin Z, Zeng Q (2021) A blockchain-enabled deduplicatable data auditing mechanism for network storage services. IEEE Trans Emerg Top Comput 9(3):1421–1432. https://doi.org/10.1109/TETC.2020.3005610
40. Qi L, Liu Y, Zhang Y, Xu X, Bilal M, Song H (2022) Privacy-aware point-of-interest category recommendation in internet of things. IEEE Internet Things J 9(21):21398–21408
41. Li J, Peng H, Cao Y, Dou Y, Zhang H, Yu P, He L (2021) Higher-order attribute-enhancing heterogeneous graph neural networks. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE20213074654
42. Zhou J, Li L, Vajdi A, Zhou X, Wu Z (2021) Temperature-constrained reliability optimization of industrial cyber-physical systems using machine learning and feedback control. IEEE Trans Autom Sci Eng. https://doi.org/10.1109/TASE.2021.3062408
43. Zhou X, Liang W, Kevin I, Wang K, Yang LT (2020) Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations. IEEE Trans Comput Soc Syst 8(1):171–178
44. Xie Y, Sheng Y, Qiu M, Gui F (2022) An adaptive decoding biased random key genetic algorithm for cloud workflow scheduling. Eng Appl Artif Intell 112:104879
45. Liu Y, Li D, Wan S, Wang F, Dou W, Xu X, Li S, Ma R, Qi L (2022) A long short-term memory-based model for greenhouse climate prediction. Int J Intell Syst 37(1):135–151
46. Zhou X, Yang X, Ma J, Kevin I, Wang K (2021) Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2021.3077937
47. Zhou J, Zhang M, Sun J, Wang T, Zhou X, Hu S (2022) DRHEFT: Deadline-Constrained Reliability-Aware HEFT algorithm for real-time heterogeneous MPSoC systems. IEEE Trans Reliab 71(1):178–189
48. Yang Y, Yang X, Heidari M, Khan MA, Srivastava G, Khosravi M, Qi L (2022) Astream: Data-stream-driven scalable anomaly detection with accuracy guarantee in IIoT environment. IEEE Trans Netw Sci Eng. https://doi.org/10.1109/TNSE20223157730
49. Xu Y, Ren J, Zhang Y, Zhang C, Shen B, Zhang Y (2019) Blockchain empowered arbitrable data auditing scheme for network storage as a service. IEEE Trans Serv Comput 13(2):289–300
50. Dai H, Xu Y, Chen G, Dou W, Tian C, Wu X, He T (2022) Rose: Robustly safe charging for wireless power transfer. IEEE Trans Mob Comput 21(6):2180–2197
51. Huang J, Tong Z, Feng Z (????) Geographical poi recommendation for internet of things: A federated learning approach using matrix factorization. Int J Commun Syst. https://doi.org/10.1002/dac.5161
52. Priyadarshini R, Quadir MdA, Rajendran N, Neelanarayanan V, Sabireen H (2022) An enhanced encryption-based security framework in the cps cloud. J Cloud Comput 11(1):1–13
53. Li Y, Liao C, Wang Y, Wang C (2015) Energy-efficient optimal relay selection in cooperative cellular networks based on double auction. IEEE Trans Wirel Commun 14(8):4093–4104
54. Xue X, Wang S, Zhang L, Feng Z, Guo Y (2018) Social learning evolution (sle): computational experiment-based modeling framework of social manufacturing. IEEE Trans Ind Inform 15(6):3343–3355

Publisher's Note