Journal of Information Hiding and Privacy Protection
DOI:10.32604/jihpp.2020.016299
Article

Random Forests Algorithm Based Duplicate Detection in On-Site Programming Big Data Environment

Qianqian Li[1], Meng Li[2], Lei Guo[3,*] and Zhen Zhang[4]

[1]University of Science and Technology Beijing, Beijing, 100083, China
[2]Beijing University of Posts and Telecommunications, Beijing, 100876, China
[3]Systems Engineering Institute AMS, Beijing, 100071, China
[4]Audio Analytic, 2 Quayside, Cambridge, CB5 8AB, UK
[*]Corresponding Author: Lei Guo. Email: guolei_jk@163.com
Received: 18 December 2020; Accepted: 02 January 2021

Abstract: On-site programming big data refers to the massive data generated in the process of software development with the characteristics of real-time, complexity and high-difficulty for processing. Therefore, data cleaning is essential for on-site programming big data. Duplicate data detection is an important step in data cleaning, which can save storage resources and enhance data consistency. Due to the insufficiency in traditional Sorted Neighborhood Method (SNM) and the difficulty of high-dimensional data detection, an optimized algorithm based on random forests with the dynamic and adaptive window size is proposed. The efficiency of the algorithm can be elevated by improving the method of the key-selection, reducing dimension of data set and using an adaptive variable size sliding window. Experimental results show that the improved SNM algorithm exhibits better performance and achieve higher accuracy.

Keywords: On-site programming big data; duplicate record detection; random forests; adaptive sliding window

1 Introduction

A surge in the amount of data produced by programming is caused by the rapid growth of software industry because thousands of companies are turning to be intelligent and digital. On-site programming big data, which refers to the massive data set that emerges instantaneously during the software programming process, and urgently needs more advanced and more efficient methods to improve programming efficiency and software quality. It covers a great variety or scope of the data points. In the future, it will construct an intelligent software development environment and architecture, with the decrease of the pressure on developers, and will achieve the purpose of man-machine collaboration.

The data cleaning of on-site programming big data is an essential step in data processing. The aim of data cleaning is to standardize and rationalize dirty data, make up the deficiency in the data, and obtain a data set that is suitable for analysis [1,2]. Data cleaning includes detecting duplicate data, processing missing value data, processing abnormal data, etc. Duplicate data will occupy the storage resources of the server, affect the quality of the data set, so it is extremely important to detect and clean similar or duplicate data in massive data [3].

The project studied in this article is duplicate data detection. The Sorted Neighborhood Method (SNM) is the commonly used method for duplicate data detection [4]. First, a key is created for sorting, which is the attribute string extracted from the record and represents a similar or duplicate record. After sorting the data set, the duplicate data will be arranged in near positions. When a suitable window size is

set, the data with high similarity will be detected in the window. The final data set will be a clean data set without redundancy after data cleaning.

However, SNM algorithm needs to compare all records included in the data set, which causes low time efficiency and accuracy. This paper proposes a random forests based approach with the dynamic and adaptive window size, which reflects on different strengths and outperform SNM. With the heavy dependence of key-selection for traditional SNM, and the fixed window size results in low efficiency, we present optimization for the feature-selection, key-creation and window size of SNM.

The feature attributes are selected by random forests to decrease data dimensions. A key is selected by vowels and consonants and the window size is decided by the similarity between fields for duplicated records detection. The main contributions of this article are as follows:

1) Aiming at the shortcomings of the traditional SNM algorithm, this paper proposes an improved algorithm adopted adaptive and dynamic size of window. We create a skeleton key for key-creation to increase the probability of detecting maximum duplicates.
2) The method of random forests is used in the selection of feature attributes, which can significantly improve the efficiency of the algorithm. It is more suitable for feature extraction in a big data environment.

The remaining of the paper is structured as follows. Section 2 presents a review of related works in literature. In Section 3, the workflow of SNM and the detailed designs of the algorithm proposed by this paper is introduced. Section 4 gives the experimental results. Finally, Section 5 presents the concluding.

2 Related Work

There is much research on data cleaning for dealing with unreasonable, incomplete and inaccurate records [5]. Duplicate record detection is an essential work of data cleaning. SNM algorithm, Multi-pass Sorted Neighborhood (MPN) algorithm and K-Nearest Neighbor algorithm are common methods for duplicate records detection problem to accomplish data cleaning [6].

The author in [7] tended to solve duplicate detection in more complex hierarchical structures, which deployed a Bayesian network to detect two duplicated data. A novel method for XML duplicate detection was proposed which only needs three elements provided by users. Reference [8] used progressive duplicate detection algorithms that can significantly increase the efficiency of finding duplicates. One of the algorithms called progressive sorted neighborhood method (PSNM) was improved from the classic SNM by sorting the input data using a predefined sorting key and only comparing records that are within a window of records in the sorted order. In addition, reference [9] made an innovation through combining two methods with PPSM and Map Reduce that work in Hadoop environment to remove similar data.

In order to solve the defects of SNM, an improved dynamic fault-tolerant algorithm was proposed to improve the efficiency of checking the effective weight and ensure the accuracy of similar duplicate records detection [6]. Jin et al. presented a nearest neighbor sort algorithm based on cluster index which adopted the combination of keywords, the cluster index, the slide window of real-time changes, the field weights and the similarity thresholds to increase the detection efficiency for alike records [5]. In addition, Reference [10] applied an optimized algorithm based on SNM by using the sliding window with variable size and speed to avoid missing record comparisons and speed up the detection.

The authors [11] offered a new idea with artificial intelligence to duplicate record detection. The genetic neural network (GNN) based approach was introduced in order to learn an optimal neural network architecture with the purpose of accelerating convergence and getting rid of the extrema-trapping dilemma. A hybrid approach called Enhanced Duplicate Count strategy was presented in [12]. The algorithm decreased comparison times and minimize the similarity distance between records to acquire higher chances to match some additional duplicates. Results displayed that the algorithm can achieve great improvements in terms of number of comparisons, precision, as well as F-Score.

A few of the related work above considers the problem of data dimensions, and a key should be considered carefully, so it is essential to provide an optimized algorithm for improving the efficiency of detection.

3 Improved SNM Algorithm Based on Random Forests

3.1 The Workflow of SNM Algorithm

Duplicate data refers to data that has duplicate content or expresses the same thing through different expressions. Because it occupies storage resources and affects the quality of the data set, duplicate data cleaning must be performed during preprocessing.

As a commonly used algorithm for repeated data detection, SNM algorithm has two flaws. One is that the implementation of SNM algorithm depend on key-selection, and the other is that the fixed size window leads to inefficient technique. The process of duplicate data detection includes three parts: Key-selection, sorting, and windowing [8,13].

1) Key-selection: Create a key as sorting keywords in the data set. The key should represent the same or similar records. The accuracy of identifying duplicate data or similar data is affected by the key selection.
2) Sorting: Create an index based on the data records and sort the data set. The same or similar records will be placed in near areas through this operation, which is helpful for the next step.
3) Windowing: The window is created to reduce the number of comparisons. The typical method of pairwise comparison is pretty complicated. By establishing a window, comparing records in the window could quickly find duplicate records, then we can increase the efficiency of data processing.

3.2 Random Forests for Features-Selection

Random forests are an integration of many classification trees. For classification problems, single classification trees are a classifier. The rules of growing random forests are as follows:

1) Use the bootstrap method to randomly select K sample sets from the provided data set to form K classification regression trees, and the rest is formed as K out-of-bag data (OOB).
2) Assume there are N feature attributes, randomly extract M features from each node of trees, calculate the information gain or Gini value of the attributes, and choose the one with the strongest classification ability to split nodes.
3) There is no restriction with the maximum growth of each tree.
4) Random forests will be formed by the generated trees, and the set of feature attributes will be classified according to the number of votes in each tree.

Random forests can be applied to massive data set, and process thousands of input samples that own high-dimensional features. Important features will be selected with record comparison times reduction and efficiency improvement.

3.3 The Process of Improved SNM Algorithm

We adopt important features acquired from random forests to choose sorted lists. The records will be processed by the improved SNM. A key is created by selecting vowels and consonants that can easily place similar records in a neighboring region. The first letter of the string, the remaining unique consonants in order of appearance and unique vowels in order of appearance are selected to compose a skeleton key. It helps to quickly find duplicate records and improve execution time. Minimum Edit Distance (MED), also called Levenshtein Distance, is an approach for calculating text similarity. More specifically, MED refers to the minimum value of the editing operations required to convert one word to another between two words. In the algorithm, it is applied to compute the distance that a string needs to be edited for being another string, the operation including replacement, deletion, and insertion. It can be

used to calculate the distance of two records. The two records are S, T respectively, and m, n are separately the length of the records. We assume that $D_{i,j}=D(S_1 \cdots S_i, T_1 \cdots T_j)$, $0 \le i \le m, 0 \le j \le n$, then we get a matrix as below:

$$D_{i,j} = \begin{cases} 0 & i,j = 0 \\ \max(i,j) & i \ or \ j = 0 \\ \min(D_{i-1,j}+1, D_{i,j-1}+1, D_{i-1,j-1}+(if \ S_i = T_i \ then \ 0 \ else \ 1)) & i,j > 0 \end{cases} \quad (1)$$

The minimum distance is d_{min}, so the similarity between the two records is as follows:

$$Sim(S,T) = 1 - \frac{d_{\min}}{\max(m,n)} \quad (2)$$

The dynamic size of window is changed according to the similarity of records in the window. We set the minimum value ω_{min} and the maximum value ω_{max} as the limitation of the window size, in order to prevent the window size from expanding too large or shrinking too small. A low similarity comparison threshold φ to adjust the degree of change of the dynamic window is given. R_1 represents the first record of the current window, and R_n represents the last record of the current window. The next window size is recorded as ω_i, the dynamic window size can be computed as follows:

$$\omega_i = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \frac{Sim(R_1, R_n)}{\varphi} \quad (3)$$

The pseudo code is provided as follows:

Algorithm: Random Forests based Sorted Neighborhood Method

input: D, φ

output: D'

```
Start
Gain the feature attributes by random forests
while(key) do
    Sorting D with key, ωi=ωmin=10, ωmax=length of D;
    Slide the window from the first position of D to start sliding;
    while(ωi does not slide to the end of D) do
        Initialize n=start of D;
        while(n<|ωi|) do
            Compare the new record entered sliding window with the first record;
            if(Similarity of R1, Rn< φ) then
            Return the index and update the window ωi;
            end
        n=n+1;
        end
        Sliding the window down;
    end
    Create clusters of index;
Compare the index of each attribute generated
if(the same index )then
Create clusters of similar records;
End
```

4 Experiment of the Improved SNM

Our evaluation is executed on the experimental data which comes from the Lending Club Loan Data Kaggle competition. The data set includes 90 features which contain numbers, dates and characters with strong discrimination of fields. We choose 2000, 4000, 8000, 12000, 15000 records separately to conduct

the experiments and 30 important feature attributes form random forests. 3% duplicate data is introduced respectively in the data to verify our algorithm. There are missing values and many different attributes in the data set, so the data set is preprocessed. We use precision rate and recall rate to evaluate the performance of the algorithm.

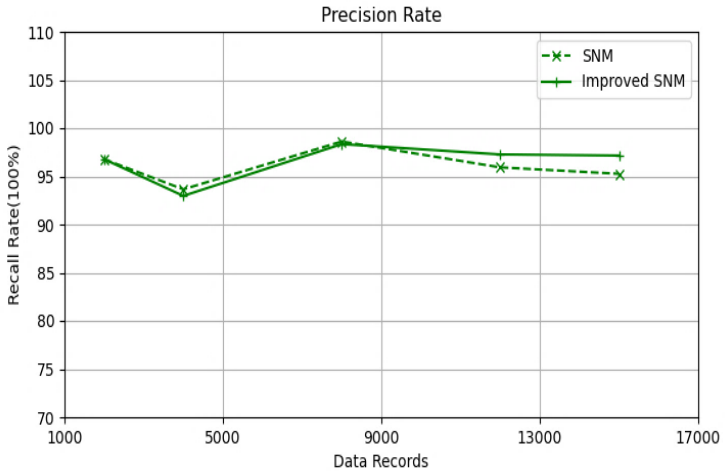

Figure 1: Precision rate

We set the window size of SNM to 10 and the low threshold to 0.65. Experiment is operating in Inter(R) Core(TM) i5-8265U CPU @1.6 GHz, Windows 10, the memory is 8 GB and the software used for experimental setup is Anaconda 3, python. Comparisons between SNM and the improved SNM in precision rate, recall rate, and execution time (Figs. 1–3) show that the improved SNM expresses the better performance.

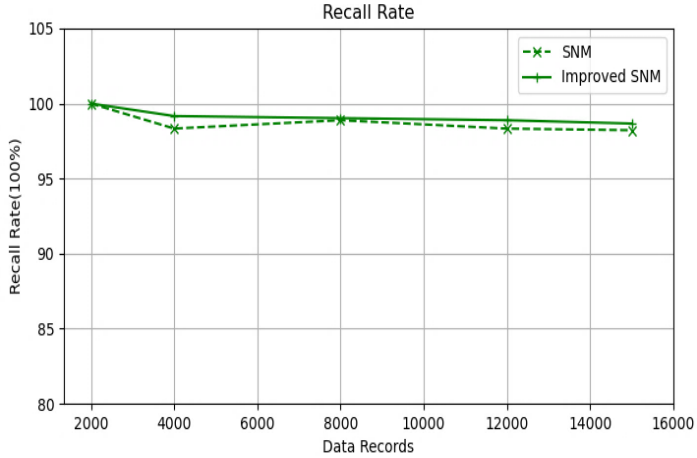

Figure 2: Recall rate

Results above show that the algorithm which adopt random forests and the sliding window with variable size can improve efficiency of detection. From the comparison of precision and recall rate, the accuracy of detection is also slightly improved.

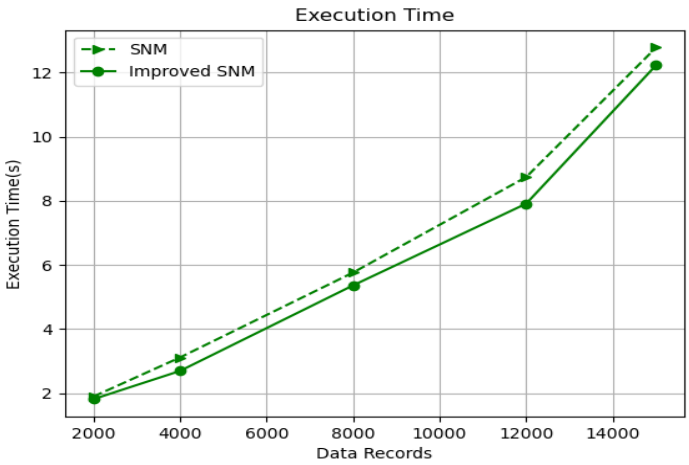

Figure 3: Execution time

5 Conclusion

Duplicate data will occupy the memory of the server and consume system resources which is more serious in on-site programming big data environment. The existence of more efficient detection algorithms is essential to save system resource.

The SNM algorithm based on random forests with the adaptive size window is proposed in this paper. Random forests are adopted to form a subset includes some representative feature attributes. Key-selection adopted in the algorithm is to create a skeleton key through the combination of special vowels and consonants, which can lead to higher matching quality. By this way, efficiency is improved by less times for data comparison. To overcome the shortcomings of SNM, we introduce dynamic and adjustable window which can reduce comparison time between two records. Experiments for duplicate data detection have proved the effectiveness and advantages of optimized algorithm proposed. The great achievement of high-dimensional massive data detection depends on the improvement of the algorithm in the on-site programming big data environment, our next efforts should be used to realize the cluster and improve the running speed of the algorithm.

Funding Statement: This work is supported by the National Key R&D Program of China (Nos. 2018YFB1003905) and the National Natural Science Foundation of China under Grant No. 61971032, Fundamental Research Funds for the Central Universities (No. FRF-TP-18-008A3).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] G. V. Dhivyabharathi and S. Kumaresan, "A survey on duplicate record detection in real world data" in *Int. Conf. on Advanced Computing & Communication Systems*, IEEE, 2016.

[2] H. L. Li, C. C. Zhou, H. T. Xu, X. Lv and Z. Han, "Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment", *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10214–10226, 2020.

[3] A. K. Elmagarmid, P. G. Ipeirotis and V. S. Verykios, "Duplicate record detection: A survey," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1–16, 2007.

[4] N. Medidar, M. Chavan, "Data duplicate detection" in *2018 9th Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT)*, 2018.

[5] H. Jin and J. Qin, "A nearest neighbor sort algorithm based on cluster index for flight support task," in *2017 29th Chinese Control and Decision Con. (CCDC),* Chongqing, pp. 2232-2237, 2017.

[6] N. N. Zhang, A. Z. Guo and T. Sun, "Research on data cleaning method based on SNM algorithm", 2017.

[7] L. Leitão, P. Calado and M. Herschel, "Efficient and effective duplicate detection in hierarchical data," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1028–1041, 2013.

[8] T. Papenbrock, A. Heise and F. Naumann, "Progressive duplicate detection," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1316–1329, 2015.

[9] B. Bhoi, P. Vyawahare, P. Avhad and N. Patil, "Data duplication avoidance in larger database," in *2017 4th Int. Conf. on Innovations in Information, Embedded and Communication Systems*, 2017.

[10] M. Li, Q. Xie and Q. Ding, "An improved data cleaning algorithm based on SNM" in *Int. Conf. on Cloud Computing and Security,* Springer International Publishing, 2015.

[11] H. R. Lu, X. Chen, X. H. Lan and F. Zheng, "Duplicate data detection using GNN," in *2016 IEEE Int. Conf. on Cloud Computing and Big Data Analysis (ICCCBDA)*, Chengdu, 2016, pp. 167–170.

[12] Y. Aassem, I. Hafidi, N. Aboutabit, "Enhanced duplicate count strategy: Towards new algorithms to improve duplicate detection" in *NISS2020: The 3rd Int. Conf. on Networking, Information Systems & Security*, 2020.

[13] J. Song, Q. Yu and R. Bao, "The detection algorithms for similar duplicate data," in *2019 6th Int. Conf. on Systems and Informatics (ICSAI)*, Shanghai, China, 2019, pp. 1534–1542.