scientific reports

Check for updates

OPEN

Intelligent retrieval method for power grid operation data based on improved SimHash and multi-attribute decision making

Songyan Zhao[1]✉, Xiaoli Guo[1], Zhaoyang Qu[1], Zhengming Zhang[1] & Tong Yu[2]

IN the trend of energy revolution, power data becomes one of the key elements of the power grid. And an advance power system with "electric power + computing power" as the core has become an inevitable choice. However, the traditional search approach based on directory query is commonly used for power grid operation data in domestic and international. The approach fails to effectively meet the user's need for fast, accurate and personalized retrieval of useful information from the vast amount of power grid data. It seriously affects the real-time availability of data and the efficiency of business-critical analytical decisions. For this reason, an intelligent retrieval approach for power grid operation data based on improved SimHash and multi-attribute decision making is proposed in this paper. This method elaborates the properties of SimHash and multi-attribute decision making algorithms. And an intelligent parallel retrieval algorithm MR-ST based on MapReduce model is designed. Finally, real time grid operation data from multiple sources are analyzed on the cloud platform for example. The experimental results show the effectiveness and precision of the method. Compared with traditional methods, the search accuracy rate, search completion rate and search time are significantly improved. Experiments show that the method can be applied to intelligent retrieval of power grid operation data.

In the trend of energy revolution, power data becomes one of the key elements of the power grid. And an advance power system with "electric power + computing power" as the core has become an inevitable choice. Power grid data has the characteristics of large size, various types, low value density, fast processing speed, etc. In recent years, large amount of data is collected and processed by automation systems and devices such as smart substations, supervisory control and data acquisition (SCADA), wide area measurement systems (WAMS) and smart meters[1]. Currently, various parts of the power system are the source of generating big data on grid. For example, on the generation side, flexible DC as the representative of the new transmission technology, new power equipment, so that the grid to collect more complex structure and different sources of data. On the electricity consumption side, along with the generation of integrated energy supply and vehicle-network convergence, interactive demand response between power companies and customers under Internet aggregation is rapidly developing. As a result, it is possible to obtain more accurate and massive power data than before[2–4]. In the background of the dramatic increase in grid operation data, the traditional retrieval method of catalog query has the disadvantages of few available resources, information overload, low accuracy, low completeness, low efficiency and slow retrieval speed. Traditional methods are no longer sufficient to meet the needs of companies to obtain data efficiently.

After decades of rapid development, the power system has long become a large interconnected system. The system consists of the following components: high-temperature, high-pressure, supercritical and ultra-supercritical units, large-capacity long-distance transmission networks, and real-time changing loads, etc. Data from multiple platforms and sources constitute the big data of electricity, which makes the new power system operation data rise dramatically. This data contains a wealth of value for society and the power system. Analysis and mining of relevant power data helps power systems to quickly retrieve matching information and facilitate advance scientific decision-making. However, the real-time operation data of the power grid has the characteristics of

[1]School of Computer Science, Northeast Electric Power University, Jilin, China. [2]Electric Power Research Institute, Guangxi Power Grid Co., Ltd, Nanning, China. ✉email: 1032322086@qq.com

time, localization and accumulation. These characteristics cause the data retrieval methods to have difficulties in retrieval, low retrieval efficiency and other problems. Therefore, how to achieve efficient retrieval of grid operation data in a large-scale data size environment is an important challenge[5].

To address these issues, this paper dissects algorithms such as SimHash and multi-attribute decision making based on the MapReduce programming model[6]. In this paper, an algorithm MR-ST is proposed for grid operation data with parallel intelligent retrieval. The proposed algorithm extracts feature from the ontology database and maps them to a unique retrieval feature vector. The retrieved vector will be converted to an intelligent search of its feature vector. Finally, example analysis of multi-source real-time grid operation data on Hadoop platform. The experimental results show that.

(a) The algorithm proposed in this paper can efficiently and accurately retrieve the retrieval results of large amount of grid operation data, and the search accuracy rate, search completion rate and retrieval time are significantly improved compared with traditional methods.
(b) The algorithm not only effectively solves the problems of intelligent retrieval of large-scale grid operation data and rational utilization of software and hardware resources, but also changes the dependence of traditional methods on high-dimensional data.

The proposed approach is applicable to the intelligent retrieval of grid operation data, and it can effectively improve the efficiency of data acquisition and mining of key business analysis and decision making for each power company in real-time.

Traditional SimHash algorithm and its improvement

The traditional SimHash algorithm calculates the similarity of the original vector by generating a fingerprint. The more similar the original data, the more similar the fingerprint obtained. This method identifies the degree of difference of each content on the bit[7]. The SimHash algorithm steps are shown below[6](1) Initialize the vector F (f dimensions) to 0 and the SimHash fingerprint S (f bits) to 0.
(2) Extracts the keywords inputed by the user into the retrieved document and calculates their weights.
(3) The same Hash function is used to calculate the fingerprint b (f bits) of the keyword, and each bit in b is scanned one by one. If in b, its i-th position is 1, then in F, the value of its i-th position plus the weight of the related keyword. Otherwise, subtract the weight of related keywords.
(4) In F, if the value on the i-th position of F is positive, then let the value on the i-th position in S be 1. Otherwise, set the value to 0, and output the fingerprint S.
(5) The Hemming distance of the fingerprint S is compared and then compared to the specified threshold. Finally, the comparison result is used to determine whether the vectors are similar or not.

SimHash algorithm has many flaws. For keywords, there is a lower search efficiency and accuracy of the extraction problem and there is also the case of inefficient matching of retrieval. In addition, it also appears that the retrieval accuracy of the weight calculation is not refined. Therefore, for the current problem of feature extraction of power grid operation data, this paper uses NLPIR PARSER Chinese word separation tool for Chinese word separation. The tool is capable of new word discovery, adaptive word splitting and keyword extraction[8]. The resulting keywords $t_{(i,n)}$ are obtained and the speed and accuracy of keyword calculation is improved. The lexical weights $w_{1(i,j)}$ of keywords are calculated by hierarchical analysis, where the weight of nouns is set to 0.3, verbs are set to 0.2, etc.[9]. Furthermore, the traditional word frequency-inverse document frequency TF-IDF algorithm is used to calculate the word frequency weight $w_{2(i,j)}$ of keyword $t_{(i,n)}$[8,10]. On this basis, the algorithm adds new word weights $w_{3(i,j)}$ and word span weights $w_{4(i,j)}$, and finally derives the weights $W_{(i,n)}$ for keyword $t_{(i,n)}$. The relevant definitions are as follows

Definition 1 Term frequency TF.

TF (Term Frequency) indicates: how often a keyword appears in the search database. Therefore, the definition of keyword frequency TF for keyword A is given by the following equation.

$$tf_{i,j} = \frac{n_{i,j}}{\sum\limits_{Q} n_{i,j}} \tag{1}$$

where $n_{i,j}$ is the frequency of occurrence of $t_{(i,n)}$ in the retrieved database D_j. Denominator $\sum\limits_{Q} n_{i,j}$ is the total number of occurrences of all words in the search database D_j.

Definition 2 Inverse Document Frequency IDF.

IDF (Inverse Document Frequency) indicates: the rating of the importance of a word as a keyword. Therefore, the definition of the inverse document frequency IDF for a keyword $t_{(i,n)}$ is as follows.

$$idf_i = \lg(N/n_{i,j} + \alpha) \tag{2}$$

where N is denoted as the total number of words in the database; $n_{i,j}$ is the total number of keywords $t_{(i,n)}$ in the database; α is an empirical value, generally taken as 0.01, 0.1, 1.

Definition 3 Traditional TF-IDF.

(1) Find the word frequency weight of keyword $t_{(i,n)}$, defined as follows.

$$w_{2(i,j)}=tf_{i,j} \times idf_i = \frac{n_{i,j}}{\sum\limits_{Q} n_{i,j}} \times \lg\left(\frac{N}{n_{i,j}}+\alpha\right) \tag{3}$$

(2) Find the new word weight $w_{3(i,j)}$ for keyword $t_{(i,n)}$, defined as follows.

$$w_{3(i,j)} = \frac{\frac{n_{i,j}}{\sum\limits_{Q} n_{i,j}} \times \lg\left(\frac{N}{n_{i,j}}+\alpha\right) + len(t)}{\sqrt{\sum\limits_{p=1}^{\beta}\left[\frac{n_{i,j}}{\sum\limits_{Q} n_{i,j}} \times \lg\left(\frac{N}{n_{i,j}}+\alpha\right)\right]^2 + len(t)}} \tag{4}$$

where **Len(t)** refers to the length of the new word; β represents an empirical value, and its value can be randomly selected. In general, in the appropriate value, any β value can be taken by cross-validation with reference to the sample.

(3) Find the word span weight $w_{4(i,j)}$ for keyword $t_{(i,n)}$, defined as follows.

$$w_{4(i,j)} = \frac{l_i}{L} \tag{5}$$

where l_i refers to the number of paragraphs in which the keyword appears and **L** indicates the total number of paragraphs.

Definition 4 Improved TF-IDF.

The weight $W_{(i,n)}$ of keyword $t_{(i,n)}$ is defined as follows

$$\begin{aligned} W_{(\mathrm{i.j})} &= w_{1_{(i.j)}} \times w_{2_{(i.j)}} \times w_{3_{(i.j)}} \times w_{4_{(i.j)}} \\ &= w_{1_{(i.j)}} \times [\frac{n_{i,j}}{\sum\limits_{Q} n_{i,j}} \times \lg(\frac{N}{n_{i,j}}+\alpha)] \times \frac{\frac{n_{i,j}}{\sum\limits_{Q} n_{i,j}} \times \lg(\frac{N}{n_{i,j}}+\alpha) + len(t)}{\sqrt{\sum\limits_{p=1}^{k}[\frac{n_{i,j}}{\sum\limits_{Q} n_{i,j}} \times \lg(\frac{N}{n_{i,j}}+\alpha)]^2 + len(t)}} \times \frac{l_i}{L} \end{aligned} \tag{6}$$

Multi-attribute decision making algorithm

The multi-attribute decision making algorithm TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) is used to calculate the utility values of grid operation data based on the improved SimHash[11,12]. The method can filter the retrieved grid operation data in a targeted way. The TOPSIS uses the "ideal solution" and "negative ideal solution" of the multi-attribute decision making method to rank them one by one and determine the combined performance of each search solution. If there is a solution which is the closest to the ideal solution and the farthest from the negative ideal solution, it is the optimal solution retrieved among **k** retrieval solutions. Expert knowledge can be embedded into the system in advance to provide objective scoring of grid operation data from various factors (e.g. weather, peak periods, etc.) at different times and environments. The process of comparing and evaluating grid operation data based on the TOPSIS method in the data retrieval process is as follows

(1) The method uses grid operation data performance characteristics as indicators for grid data evaluation and retrieval of qualitative requirements to create an initial decision matrix $\mathbf{VF}_k$.

   (1) Similarity retrieval vector $\mathbf{V}_k = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_k]$ of retrieved grid operation data
   (2) Set $\mathbf{B}_k = [\mathbf{b}_1 \ \mathbf{b}_2 \ \cdots \ \mathbf{b}_k]$ to be a vector of retrieved qualitative demand indicators
   (3) The initial matrix can be expressed as $\mathbf{VF}_k = (\mathbf{vf}_{ij})_{m\times n}$.

   Where $\mathbf{vf}_{ij}$ denotes the score value of its j th indicator in the i th data

(2) Normalization of the decision matrix

$$\mathrm{vf}_{ij}^{*} = \mathrm{vf}_{ij} \Big/ \sum_{i=1}^{k} \mathrm{vf}_{ij}^{2} \tag{7}$$

   The normalized decision matrix is $\mathbf{VF}_k^{*} = (vf_{ij}^{*})_{\mathrm{m}\times n}$.

(3) Normalize the retrieved qualitative demand vector $\mathbf{B}_k = [\mathbf{b}_1 \ \mathbf{b}_2 \ \cdots \ \mathbf{b}_n]$. The normalized demand vector is $\mathbf{B}_k^{*} = [\mathbf{b}_1^{*} \ \mathbf{b}_2^{*} \ \cdots \ \mathbf{b}_n^{*}]$. where $\mathbf{b}_j^{*}$ represents the weight of the normalized demand indicator **j**.

$$b_j^* = b_j \Big/ \sum_{j=1}^{n} b_j \tag{8}$$

(4) Calculate the distance from each retrieved grid operation data $P_i = (1 \le i \le k)$ to the ideal solution and the negative ideal solution

$$\varphi_i^+ = \sqrt{\sum_{j=1}^{n} \left[b_j^* \left(vf_{ij}^* - vf_{j\_best}^* \right) \right]^2} \tag{9}$$

$$\varphi_i^- = \sqrt{\sum_{j=1}^{n} \left[b_j^* \left(vf_{ij}^* - vf_{j\_worst}^* \right) \right]^2} \tag{10}$$

where, n refers to the total retrieval of grid operation data on qualitative characteristics; $\mathbf{vf}_{ij}^*$ indicates the score obtained after normalizing $\mathbf{vf}_{ij}$; $\mathbf{b}_j^*$ represents the demand weight obtained after normalization of $\mathbf{b}_j$; $\mathbf{vf}_{j\_best}^*$, $\mathbf{vf}_{j\_worst}^*$ represent the **j**th subscoring value of the ideal solution and the **j**th subscoring value of the resulting negative ideal solution after normalization, respectively.

(5) Calculate the utility value U_i for all retrieved grid operation data $P_i = (1 \le i \le k)$.

$$U_i = \frac{\varphi_i^-}{\varphi_i^+ + \varphi_i^-} \tag{11}$$

Intelligent retrieval of grid operation data based on MapReduce model

Principle of MR-ST algorithm. MapReduce is proposed by Google. MapReduce has the practical effect of enabling a parallel programming model for processing massive data sets[13]. The MR-ST algorithm based on improved SimHash and multi-attribute decision techniques explains the traditional SimHash algorithm and combines it with MapReduce model, TF-IDF and multi-attribute decision making algorithms. MR-ST can effectively implement an intelligent retrieval process for parallel retrieval of grid operation data on a cloud platform.

MR-ST algorithm retrieves similar vectors of massive grid operation data in parallel based on MapReduce model. The Map function in the algorithm can autonomously complete the slicing operation of each data. Reduce function is to sort the results of multiple Map. Due to their known performance, they can be used in combination with other functions. Algorithms parallelize the processing of Map and Reduce functions, and a programming model can be formed.

Introduction to the Map function

Map function provides simultaneous access to n data slice. The function extracts the retrieved row numbers and their retrieved attribute values in combination with the retrieval requirements and then performs the mapping calculation. This procedure requires a call to the relevant function that generates a < key, value > pair.

Introduction to the Reduce function

The Reduce function receives the results from the map of each node and performs the merge calculation. And the functions store datasets in a distributed file system (HDFS).

The MR-ST algorithm intelligent retrieval process is divided into six steps and three retrieval phases.

(1) Obtain keyword retrieval vector $\mathbf{G}_K$, sequence retrieval vector $\mathbf{X}_K$, retrieval feature vector R_K.
(2) The search preprocessing yields keyword search database $\mathbf{D}_j$, sequence search database $\mathbf{D}_{j+1}$, search feature database $\mathbf{D}_{j+2}$.
(3) Improving SimHash similarity calculation, multi-attribute decision making and comprehensive score calculation.

The principle block diagram of MR-ST algorithm is shown in Figure 1.

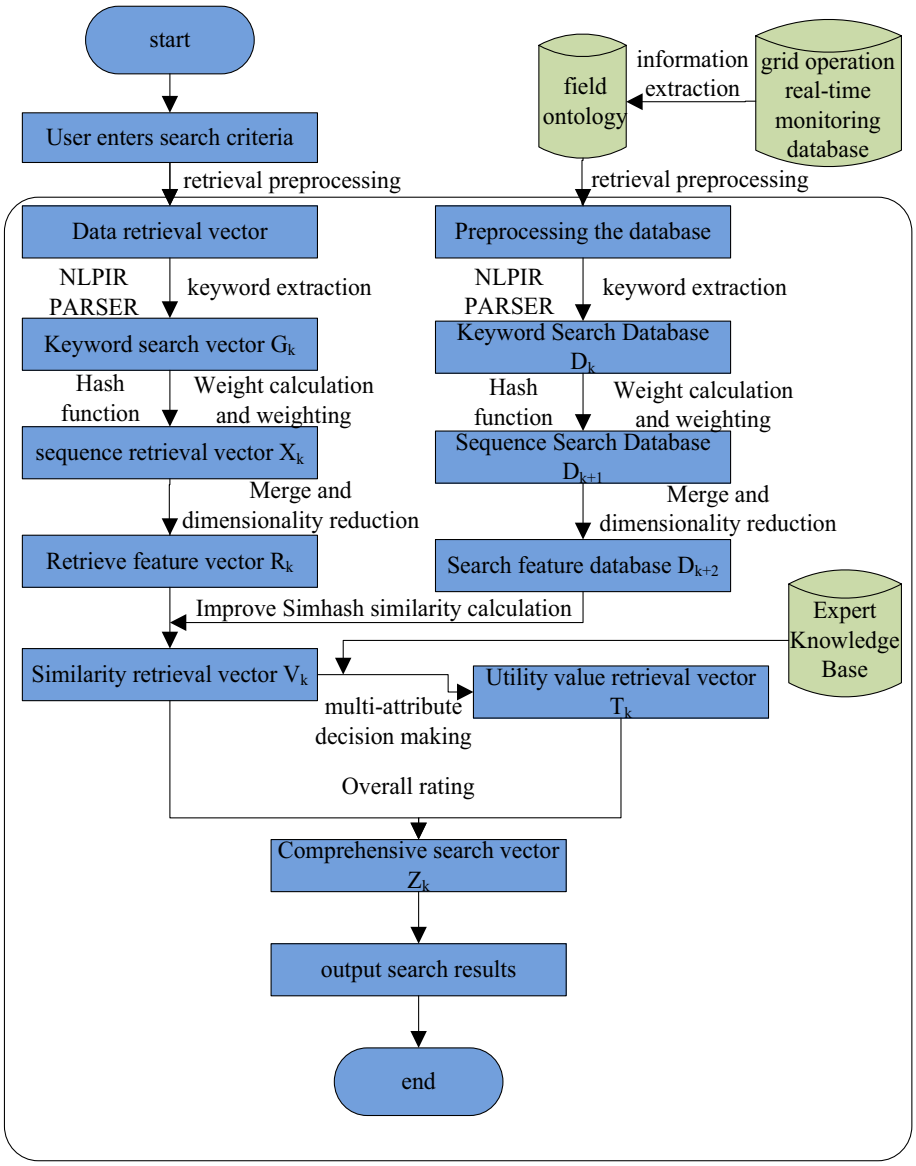


Figure 1. Principle block diagram of MR-ST algorithm.

MR-ST algorithm construction. Firstly, the paper sets the parameter symbols in the algorithm.

i. Set the database to $\mathbf{D}_k$, $K \in R$, and the row number to ID. The attribute value of column **j** of row **i** of the database is set to $V_{i,j}$ and $V_{i,j} \in V$. $t_{(i,n)}$ refers to the **n**th keyword in the **i**th row.
ii. Suppose the user inputs the search criteria data table as Y_k, $K \in R$, and the row number to id. The attribute value of column **j** of row **i** of the database is set to $E_{i,j}$ and $E_{i,j} \in E$. $m_{(i,n)}$ refers to the **n**th keyword in the **i**th row.

The MR-ST calculation process is shown below.

Process 1: Construct keyword search database D_k and keyword search vector G_K	Input: Database $\mathbf{D}_k$, user search criteria data table $\mathbf{Y}_k$ Output: Database $\mathbf{D}_k$ corresponding to the keyword search database vector D_i, the user search criteria data table Y_k corresponding to the keyword search vector G_i

Algorithm 1: Map part

Step 1: Input $\langle \text{key} = \text{ID}_i, \text{value} = \text{V}_{i,j} \rangle$, $\langle \text{key} = \text{ID}_i, \text{value} = \text{E}_{i,j} \rangle$.

Step 2: For each $\text{V}_{i,j}$ and $\text{E}_{i,j}$, the NLPIR PARSER Chinese word separation tool system was used. NLPIR PARSER splits $\text{V}_{i,j}$ and $\text{E}_{i,j}$ into different keywords $t_{(i,n)}$ and $m_{(i,n)}$.

Step 3: Sort by attribute value one by one. Using $t_{(i,n)}$ and $m_{(i,n)}$ as elements, construct the keyword search database vector $D_i = (t_{(i,n)})$ and the keyword search vector $G_i = (m_{(i,n)})$ for the search, respectively.

Step 4: Output $\langle \text{key} = \text{ID}_i, \text{value} = \mathbf{D}_i \rangle$, $\langle \text{key} = \text{id}_i, \text{value} = \mathbf{G}_i \rangle$, and store the results in HDFS.

Algorithm 2: Reduce part

Step 1: Input the results of Map

Step 2: $i = 1, 2, \cdots, size(D_k)$、$i = 1, 2, \cdots, size(Y_k)$, where $\text{size}(D_k)$ refers to all the retrieved quantities in D_k, $\text{size}(Y_k)$ refers to the total number of retrieved quantities in Y_k.

Step 3: Calculate the summation based on the key values D_i、G_i

Step 4: Output $\langle \text{key} = D_i/..., \text{value} = (\text{ID}_1, .../...) \rangle$, $\langle \text{key} = G_i/..., \text{value} = (\text{id}_1, .../...) \rangle$

The algorithm stores the output results in HDFS as keyword search database sub vectors $\mathbf{d}_k = (\text{ID}_i, \mathbf{D}_i)$, keyword search sub vectors $\mathbf{g}_k = (\text{id}_i, \mathbf{G}_i)$. If a key corresponds to multiple values of value, then value is split and corresponds to key one by one. For different searches, proximally stored in the keyword search database vector $\mathbf{d}_k$, keyword search vector $\mathbf{g}_k$. Searches with the same key in $\mathbf{d}_k$, $\mathbf{g}_k$ are each clustered into the same vector. Therefore, we get $\mathbf{D}_K = (\mathbf{d}_k)$, $\mathbf{G}_\text{k} = (\mathbf{g}_k)$.

Process 2: Construct sequence retrieval database vector $\mathbf{D}_{k+1}$ and sequence retrieval vector $\mathbf{X}_K$	Input: keyword search database vector $\mathbf{d}_k$, keyword search vector $\mathbf{g}_k$, lexical weighting table Q Output: Sequence retrieval database vector $\mathbf{D}_{k+1}$, sequence retrieval vector $\mathbf{X}_K$.

Algorithm 3: Map part

Step 1: Read the data in the HDFS file, input the keyword search database vector $\mathbf{d}_k$, keyword search vector $\mathbf{g}_k$, Algorithm 2 results $\langle \text{key} = \text{ID}_i, \text{value} = \mathbf{D}_i \rangle$, $\langle \text{key} = \text{id}_i, \text{value} = \mathbf{G}_i \rangle$.

Step 2: For each $t_{(i,n)} \in \mathbf{D}_i$, $m_{(i,n)} \in \mathbf{G}_i$, according to the word nature of $t_{(i,n)}$, $m_{(i,n)}$, set the corresponding word nature weights $w_{1(i,n)}$, $w^*_{1(i,n)}$; word frequency weights $w_{2(i,\text{n})}$, $w^*_{2(i,\text{n})}$; new word weights $w_{3(i,n)}$, $w^*_{3(i,n)}$; word span weights $w_{4(i,n)}$, $w^*_{4(i,n)}$.

$$\mathbf{w}_{1i} = (w_{1(i,n)}) = [w_{1(i,1)} \quad w_{1(i,2)} \quad \cdots \quad w_{1(i,n)}], \quad \mathbf{W}_1 = (\mathbf{w}_{1i}) = [\mathbf{w}_{1,1} \quad \mathbf{w}_{1,2} \quad \cdots \quad \mathbf{w}_{1,i}]$$

$$w_{2(i,n)} = TF - IDF(t_{(i,n)}), \quad \mathbf{w}_{2i} = (w_{2(i,n)}) = [w_{2(i,1)} \quad w_{2(i,2)} \quad \cdots \quad w_{2(i,n)}], \quad \mathbf{W}_2 = (\mathbf{w}_{2i}) = [\mathbf{w}_{2,1} \quad \mathbf{w}_{2,2} \quad \cdots \quad \mathbf{w}_{2,i}]$$

$$w_{3(i,n)} = TF - IDF(t_{(i,n)}), \quad \mathbf{w}_{3i} = (w_{3(i,n)}) = [w_{3(i,1)} \quad w_{3(i,2)} \quad \cdots \quad w_{3(i,n)}], \quad \mathbf{W}_3 = (\mathbf{w}_{3i}) = [\mathbf{w}_{3,1} \quad \mathbf{w}_{3,2} \quad \cdots \quad \mathbf{w}_{3,i}]$$

$$w_{4(i,n)} = TF - IDF(t_{(i,n)}), \quad \mathbf{w}_{4i} = (w_{4(i,n)}) = [w_{4(i,1)} \quad w_{4(i,2)} \quad \cdots \quad w_{4(i,n)}], \quad \mathbf{W}_4 = (\mathbf{w}_{4i}) = [\mathbf{w}_{4,1} \quad \mathbf{w}_{4,2} \quad \cdots \quad \mathbf{w}_{4,i}]$$

$$\mathbf{w}^*_{1i} = (w^*_{1(i,n)}) = [w^*_{1(i,1)} \quad w^*_{1(i,2)} \quad \cdots \quad w^*_{1(i,n)}], \quad \mathbf{W}^*_1 = (\mathbf{w}^*_{1i}) = [\mathbf{w}^*_{1,1} \quad \mathbf{w}^*_{1,2} \quad \cdots \quad \mathbf{w}^*_{1,i}]$$

$$w^*_{2(i,n)} = TF - IDF(m_{(i,n)}), \quad \mathbf{w}^*_{2i} = (w^*_{2(i,n)}) = [w^*_{2(i,1)} \quad w^*_{2(i,2)} \quad \cdots \quad w^*_{2(i,n)}], \quad \mathbf{W}^*_2 = (\mathbf{w}^*_{2i}) = [\mathbf{w}^*_{2,1} \quad \mathbf{w}^*_{2,2} \quad \cdots \quad \mathbf{w}^*_{2,i}]$$

$$w^*_{3(i,n)} = TF - IDF(m_{(i,n)}), \quad \mathbf{w}^*_{3i} = (w^*_{3(i,n)}) = [w^*_{3(i,1)} \quad w^*_{3(i,2)} \quad \cdots \quad w^*_{3(i,n)}], \quad \mathbf{W}^*_3 = (\mathbf{w}^*_{3i}) = [\mathbf{w}^*_{3,1} \quad \mathbf{w}^*_{3,2} \quad \cdots \quad \mathbf{w}^*_{3,i}]$$

$$w^*_{4(i,n)} = TF - IDF(m_{(i,n)}), \quad \mathbf{w}^*_{4i} = (w^*_{4(i,n)}) = [w^*_{4(i,1)} \quad w^*_{4(i,2)} \quad \cdots \quad w^*_{4(i,n)}], \quad \mathbf{W}^*_4 = (\mathbf{w}^*_{4i}) = [\mathbf{w}^*_{4,1} \quad \mathbf{w}^*_{4,2} \quad \cdots \quad \mathbf{w}^*_{4,i}]$$

Step 3: Output $\langle \text{key} = \text{ID}, \text{value} = \mathbf{W}_1 \rangle$, $\langle \text{key} = \text{ID}, \text{value} = \mathbf{W}_2 \rangle$, $\langle \text{key} = \text{ID}, \text{value} = \mathbf{W}_3 \rangle$, $\langle \text{key} = \text{ID}, \text{value} = \mathbf{W}_4 \rangle$; $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^*_1 \rangle$, $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^*_2 \rangle$, $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^*_3 \rangle$, $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^*_4 \rangle$

Step 4: In $i = 1, 2, \cdots, size(D_k)$、$j = 1, 2, \cdots, n$; $i = 1, 2, \cdots, size(Y_k)$、$j = 1, 2, \cdots, n$, when ($\langle \text{key} = \text{ID}, \text{value} = \mathbf{W}_1 \rangle$, $\langle \text{key} = \text{ID}, \text{value} = \mathbf{W}_2 \rangle$, $\langle \text{key} = \text{ID}, \text{value} = \mathbf{W}_3 \rangle$, $\langle \text{key} = \text{ID}, \text{value} = \mathbf{W}_4 \rangle$); ($\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^*_1 \rangle$, $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^*_2 \rangle$, $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^*_3 \rangle$, $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^*_4 \rangle$), execute:

$$W_{(i,n)} = w_{1(i,n)} \times w_{2(i,n)} \times w_{3(i,n)} \times w_{4(i,n)}, \quad \mathbf{w}_i = (w_{(i,n)}) = [w_{(i,1)} \quad w_{(i,2)} \quad \cdots \quad w_{(i,n)}], \quad \mathbf{W} = (\mathbf{w}_i) = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_i]$$

$$W^*_{(i,n)} = w^*_{1(i,n)} \times w^*_{2(i,n)} \times w^*_{3(i,n)} \times w^*_{4(i,n)}, \quad \mathbf{w}^*_i = (w^*_{(i,n)}) = [w^*_{(i,1)} \quad w^*_{(i,2)} \quad \cdots \quad w^*_{(1,n)}], \quad \mathbf{W}^* = (\mathbf{w}^*_i) = [\mathbf{w}^*_1 \quad \mathbf{w}^*_2 \quad \cdots \quad \mathbf{w}^*_n]$$

Step 5: Output $\langle \text{key} = \text{ID}, \text{value} = \mathbf{W} \rangle$, $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^* \rangle$

Algorithm 4: Map part

Step 1: Input the results of Algorithm 2 $\langle \text{key} = \text{ID}_i, \text{value} = \mathbf{D}_i \rangle$ 、 $\langle \text{key} = \text{id}_i, \text{value} = \mathbf{G}_i \rangle$

Step 2: The same traditional hash function is used to find the hash value of $t_{(i,n)}$ and $m_{(i,n)}$. The output method is fixed-length output. $i = 1, 2, \cdots, size(D_k)$、$j = 1, 2, \cdots, n$; $i = 1, 2, \cdots, size(Y_k)$、$j = 1, 2, \cdots, n$, $f_{(i,n)}$ =Hash($t_{(i,n)}$), $f^*_{(i,n)}$ =Hash($m_{(i,n)}$). Where $f_{(i,n)}$, $f^*_{(i,n)}$ is the **r**-dimensional vector.

$$f_i = [f_{(i,1)} \quad f_{(i,2)} \quad \cdots \quad f_{(i,n)}], \quad f = [f_1 \quad f_2 \quad \cdots \quad f_i]$$

$$f_i^* = [f^*_{(i,1)} \quad f^*_{(i,2)} \quad \cdots \quad f^*_{(i,n)}], \quad f^* = [f_1^* \quad f_2^* \quad \cdots \quad f_i^*]$$

Step 3: Output $\langle \text{key} = \text{ID}, \text{value} = f \rangle$ 、 $\langle \text{key} = \text{id}, \text{value} = f^* \rangle$

Algorithm 5: Map part

Step 1: Input the results of Algorithm 3 $\langle \text{key} = \text{ID} , \text{value} = \mathbf{W} \rangle$ 、 $\langle \text{key} = \text{id}, \text{value} = \mathbf{W}^* \rangle$

Step 2: Input the results of Algorithm 4 $\langle \text{key} = \text{ID}, \text{value} = f \rangle$ 、 $\langle \text{key} = \text{id}, \text{value} = f^* \rangle$

Step 3: Initialize the **r**-dimensional vector $\mathbf{F}_{(i,n)}$=(0)、$\mathbf{F}^*_{(i,n)}$=(0); $\mathbf{S}$=(0)、$\mathbf{S}^*$=(0)

Step 4: $i = 1, 2, \cdots, size(D_k)$、$j = 1, 2, \cdots, n$; $i = 1, 2, \cdots, size(Y_k)$、$j = 1, 2, \cdots, n$, If bit λ of $f_{(i,n)}$ and $f^*_{(i,n)}$ are greater than 0, $\max(\lambda) = r$, then add $w_{1(i,n)}$ and $w^*_{1(i,n)}$ to bit λ of $\mathbf{F}_{(i,n)}$ and $\mathbf{F}^*_{(i,n)}$. Otherwise, subtract it from $w_{1(i,n)}$ and $w^*_{1(i,n)}$.

$$F_i = [F_{(i,1)} \quad F_{(i,2)} \quad \cdots \quad F_{(i,n)}], \quad F = [F_1 \quad F_2 \quad \cdots \quad F_i]$$

$$F_i^* = [F^*_{(i,1)} \quad F^*_{(i,2)} \quad \cdots \quad F^*_{(i,n)}], \quad F^* = [F_1^* \quad F_2^* \quad \cdots \quad F_i^*]$$

Step 5: $i = 1, 2, \cdots, size(D_k)$、$j = 1, 2, \cdots, n$; $i = 1, 2, \cdots, size(Y_k)$、$j = 1, 2, \cdots, n$, $\mathbf{S}_i = \square F_{i,j}$, $\mathbf{S} = [\mathbf{S}_1 \quad \mathbf{S}_2 \quad \cdots \quad \mathbf{S}_i]$; $\mathbf{S}^*_i = \square F^*_{i,j}$, $\mathbf{S}^* = [\mathbf{S}_1^* \quad \mathbf{S}_2^* \quad \cdots \quad \mathbf{S}_i^*]$. Where $\square F_{i,j}$ and $\square F^*_{i,j}$ mean the corresponding positions of different strings $F_{i,j}$ and $F^*_{i,j}$ with the same number of bits, and the sum of them respectively.

Step 6 : Output $\langle \text{key} = \text{ID}, \text{value} = \mathbf{S} \rangle$ 、 $\langle \text{key} = \text{id}, \text{value} = \mathbf{s}^* \rangle$

Algorithm 6: Reduce part

Step 1: Input Map results $\langle \text{key} = \text{ID}, \text{value} = \mathbf{S} \rangle$ 、 $\langle \text{key} = \text{id}, \text{value} = \mathbf{s}^* \rangle$

Step 2: Subsumption calculation with $\mathbf{s}$ as key

Step 3: Output $\langle \text{key} = \mathbf{S}_i /..., \text{value} = (ID_1, .../...) \rangle$ 、 $\langle \text{key} = \mathbf{S}_i^* /..., \text{value} = (\text{id}_1, .../...) \rangle$

The results are stored in HDFS as sequence retrieval database sub vector $\mathbf{d}_{k+1} = (\text{ID}_i, \mathbf{S}_i)$ and sequence retrieval sub vector $\mathbf{x}_k = (\mathbf{ID}_i, \mathbf{S}_i^*)$, $D_{k+1} = [f_1 \, f_2 \, \cdots \, f_i]$, $\mathbf{X}_k = [f_1^* \, f_2^* \, \cdots \, f_i^*]$. If a key corresponds to multiple values of value, then value is split and corresponds to key one by one. The value are stored in the sequence search database sub vector $\mathbf{d}_{k+1}$, sequence search sub vector $\mathbf{x}_k$ in the order of ID, id from largest to smallest, respectively.

Process 3: Obtain the retrieval feature database vector $\mathbf{D}_{k+2}$ and the retrieval feature vector $\mathbf{R}_K$.	Input: Sequence retrieval database vector $\mathbf{D}_{k+1}$, sequence retrieval vector $\mathbf{X}_K$. Output: Obtain the retrieval feature database vector $\mathbf{D}_{k+2}$ and the retrieval feature vector $\mathbf{R}_K$.

Algorithm 7: Map part

Step 1: Initialize the **r**-dimensional vector, $\mathbf{S}_i = (0)$、$\mathbf{S}_i^* = (0)$.

Step 2: Input the results of Algorithm 6 $\langle \text{key} = \text{ID}_i, \text{value} = \mathbf{S}_i \rangle$、$\langle \text{key} = \text{id}_i, \text{value} = \mathbf{S}_i^* \rangle$.

Step 3: $i = 1, 2, \cdots, size(D_k)$、$j = 1, 2, \cdots, n$; $i = 1, 2, \cdots, size(Y_k)$、$j = 1, 2, \cdots, n$. If the λ th bit in $\mathbf{S}_i$, $\mathbf{S}_i^*$ are each greater than 0, then the λ th bit of $\mathbf{S}_i$ and $\mathbf{S}_i^*$ are each noted as 1. Otherwise, it is noted as 0. $\mathbf{S}_k = [\mathbf{S}_1 \quad \mathbf{S}_2 \quad \cdots \quad \mathbf{S}_i]$、$\mathbf{S}_k^* = [\mathbf{S}_1 \quad \mathbf{S}_2 \quad \cdots \quad \mathbf{S}_i]$

Step 4: Output $\langle \text{key} = \text{ID}, \text{value} = \mathbf{S}_k \rangle$、$\langle \text{key} = \text{id}, \text{value} = \mathbf{S}_K^* \rangle$, and store retrieve feature database sub vector $\mathbf{d}_{k+2} = (\text{ID}, \mathbf{S}_k)$, retrieve feature sub vector $r_k = (\text{id}, \mathbf{S}_k^*)$ in HDFS, and $\mathrm{D}_{k+2} = (d_{k+2}) = [d_1 \quad d_2 \quad \cdots \quad d_{k+2}]$、$\mathrm{R}_K = (r_k) = [r_1 \quad r_2 \quad \cdots \quad r_k]$.

Algorithm 8: Reduce part

Step 1: Input the result of Map in Algorithm 7 d_{k+2}、$\langle \text{key} = \text{ID}, \text{value} = \mathbf{S}_k \rangle$; r_k、$\langle \text{key} = \text{id}, \text{value} = \mathbf{S}_k^* \rangle$

Step 2: Calculated by merging the values $\mathbf{S}_k$ and $\mathbf{S}_k^*$.

Step 3: Output $\langle \text{key} = \mathbf{S}_i /..., \text{value} = (\text{ID}_1, .../...) \rangle$、$\langle \text{key} = \mathbf{S}_i^* /..., \text{value} = (\text{id}_1, .../...) \rangle$

In the output, the similar search vectors $\mathbf{S}_i$ and $\mathbf{S}_i^*$ are clustered together respectively. According to the number of ID and id, the corresponding similar search sub vectors d_{k+2} and r_k are stored in HDFS, and $\mathrm{D}_{k+2} = (d_{k+2}) = [d_1 \; d_2 \; \cdots \; d_{k+2}], \mathrm{R}_K = (r_k) = [r_1 \; r_2 \; \cdots \; r_k]$.

Procedure 4: Create SimHash search vector $\mathbf{DV}_k$, $\mathbf{RV}_k$	Input: Input the result of Algorithm 8 $\mathtt{D}_{k+2}$, $\mathtt{R}_K$ in HDFS. Output: SimHash search vector $\mathbf{DV}_k$, $\mathbf{RV}_k$.

Algorithm 9: Map part

Step 1: Input the data $\langle \text{key} = \text{ID}_i, \text{value} = \mathbf{S}_i \rangle$、$\langle \text{key} = \text{id}_i, \text{value} = \mathbf{S}_i^* \rangle$ in HDFS.

Step 2: Divide **r** bits of $\mathbf{S}_i$ and $\mathbf{S}_i^*$ equally into **b** parts. $\mathbf{S}_i = [\mathbf{S}_{i1}\ \mathbf{S}_{i2}\ \cdots\ \mathbf{S}_{ib}]$, $\mathbf{S}_i^* = [\mathbf{S}_{i1}^*\ \mathbf{S}_{i2}^*\ \cdots\ \mathbf{S}_{ib}^*]$. Mapping key to $\mathbf{S}_{ib}$ and $\mathbf{S}_{ib}^*$.

Step 3: Output $\langle \text{key} = \text{ID}_i, \text{value} = \mathbf{S}_{i1} \rangle$ … $\langle \text{key} = \text{ID}_i, \text{value} = \mathbf{S}_{ik} \rangle$; $\langle \text{key} = \text{id}_i, \text{value} = \mathbf{S}_{i1}^* \rangle$ … $\langle \text{key} = \text{id}_i, \text{value} = \mathbf{S}_{ik}^* \rangle$;

Algorithm 10: Reduce part

Step 1: Input the result of Map $\langle \text{key} = ID_i, \text{value} = \mathbf{S}_{ik} \rangle$, $\langle \text{key} = id_i, \text{value} = \mathbf{S}_{ik}^* \rangle$

Step 2: Calculated by merging the values key, ID_i, id_i

Step 3: Output $\langle \text{key} = ID_i, \text{value} = (\mathbf{S}_{i1}, \mathbf{S}_{i2}, ...\mathbf{S}_{ib}) \rangle$、$\langle \text{key} = id_i, \text{value} = (\mathbf{S}_{i1}^*, \mathbf{S}_{i2}^*, ...\mathbf{S}_{ib}^*) \rangle$ and store in HDFS

Algorithm 11: Map part

Step 1: Input the results $\langle \text{key} = \text{ID}_i, \text{value} = (\mathbf{S}_{i1}, \mathbf{S}_{i2}, ..., \mathbf{S}_{ib}) \rangle$, $\langle \text{key} = \text{id}_i, \text{value} = (\mathbf{S}_{i1}^*, \mathbf{S}_{i2}^*, ...\mathbf{S}_{ib}^*) \rangle$ in HDFS.

Step 2: In $(\mathbf{S}_{i1}, \mathbf{S}_{i2}, ..., \mathbf{S}_{ib})$、$(\mathbf{S}_{i1}^*, \mathbf{S}_{i2}^*, ...\mathbf{S}_{ib}^*)$, pairs $\mathbf{S}_{i1}$ and $\mathbf{S}_{i1}, \mathbf{S}_{i2}, ..., \mathbf{S}_{ib}$ are swapped in turn, $\mathbf{S}_{i1}^*$ and $\mathbf{S}_{i1}^*, \mathbf{S}_{i2}^*, ...\mathbf{S}_{ib}^*$ are swapped in turn. In this process, the fingerprint with $\mathbf{S}_{ik}$、$\mathbf{S}_{ik}^*$ $(0 < k \le b)$ as the first part are obtained separately for **b**. For example, $\mathbf{S}_2 = (\mathbf{S}_{i2}, \mathbf{S}_{i1}, ..., \mathbf{S}_{ib})$、$\mathbf{S}_2^* = (\mathbf{S}_{i2}^*, \mathbf{S}_{i1}^*, ..., \mathbf{S}_{ib}^*)$ correspond to the first $\frac{r}{b}$ bit part of the b values. The pointer is $E = (E_1, E_2, ... E_{hb})$, $E^* = (E_1^*, E_2^*, ... E_{hb}^*)$ and it maps based on ID and id.

Step 3: Output $\langle \text{key} = \text{ID}_1, \text{value} = E_1 \rangle$ … $\langle \text{key} = \text{ID}_1, \text{value} = E_b \rangle$; $\langle \text{key} = \text{id}_1, \text{value} = E_1^* \rangle$ … $\langle \text{key} = \text{id}_1, \text{value} = E_b^* \rangle$;

Algorithm 12: Reduce part

Step 1: Input the result of Map in Algorithm 11 $\langle \text{key} = \text{ID}_i, \text{value} = E_k \rangle$、$\langle \text{key} = \text{id}_i, \text{value} = E_k^* \rangle$ $(0 < k \le b)$

Step 2: Calculated by merging the values key, ID_i, id_i

Step 3: Output $\langle \text{key} = \text{ID}_1, \text{value} = (E_1, E_2, \ldots, E_b) \rangle$ … $\langle \text{key} = \text{ID}_h, \text{value} = (E_{(h-1)b+1}, ..., E_{hb}) \rangle$, $\langle \text{key} = \text{id}_1, \text{value} = (E_1^*, E_2^*, \ldots, E_b^*) \rangle$ … $\langle \text{key} = \text{id}_h, \text{value} = (E_{(h-1)b+1}^*, ..., E_{hb}^*) \rangle$ $(0 < h \le bi)$

The results are stored in HDFS as a search vector $\mathbf{DV}_k = (\mathbf{ID}_i, E_i)$, $E_i = (E_{(h-1)b+1}, ...E_{hb})$, $\mathbf{RV}_k = (\text{id}_i, E_i^*)$, $E_i^* = (E_{(h-1)b+1}^*, ..., E_{hb}^*)$. $\boldsymbol{ID}_i$, E_i, S_i; id_i, E_i^*, S_i^* correspond to each other respectively.

Procedure 5: Create similarity search vectors $\mathbf{V}_k$	Input: Input $\mathbf{DV}_k$, $\mathbf{RV}_k$ in HDFS, threshold μ Output: Similarity search vectors $\mathbf{V}_k$

Algorithm 13: Map part

Step 1: Input $\mathbf{DV}_k$ in HDFS. $\langle \text{key} = \text{ID}_i, \quad \text{value} = E_i \rangle, k \in K$.

Step 2: Input $\mathbf{RV}_k$ in HDFS. $\langle \text{key} = \text{id}_i, \quad \text{value} = E_i^* \rangle, k \in K$.

Step 3: Do an xor operation on the first $r(1-\frac{\mu}{b})$ bit of both E and E^*.

Step 4: If the quantity of 1 in the data is less than μ, then output $\langle \text{key} = \text{ID}_i, \text{value} = 1 \rangle$.

Algorithm 14: Reduce part

Step 1: Input Map results $\langle \text{key=ID}_i, \text{value} = 1 \rangle$.

Step 2: Merge calculation with key as 1.

Step 3: Output $\langle \text{key} = 1, \text{value} = (\text{ID}_1, \text{ID}_2, \ldots) \rangle$

The vectors that can be clustered together in the output are the vectors with similarity. According to ID, its corresponding retrieval is stored in HDFS as similarity retrieval sub vector $\mathbf{v}_K$, and the similarity retrieval vector is $\mathbf{V}_k = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_k]$.

Procedure 6: Calculate the utility value retrieval vector $\mathbf{T}_k$	Input: Input the result similarity retrieval vector $\mathbf{V}_k$ of algorithm 14 in HDFS Output: The utility value retrieval vector $\mathbf{T}_k$

Algorithm 15: Map part

Step 1: Input the similarity search vector $\mathbf{V}_k$ from HDFS, $\langle \text{key=ID}_i, \text{value} = 1 \rangle$.

Step 2: Multi-attribute decision utility value calculation for similarity retrieval vectors $\mathbf{V}_k$.

Step 3: Output $\langle \text{key} = \text{ID}_i, \text{value} = \text{U}_i \rangle$.

Algorithm 16: Reduce part

Step 1: Input Map results $\langle \text{key} = \text{ID}_i, \text{value} = \text{U}_i \rangle$.

Step 2: Merge calculation with key as U_i.

Step 3: Output $\langle \text{key} = \text{U}_i, \text{value} = (\text{ID}_1, \text{ID}_2, \ldots) \rangle$

In the output, the utility value retrieval vectors $\mathbf{T}_k$ are clustered together. According to ID, sort according to utility values in turn and store them in HDFS.

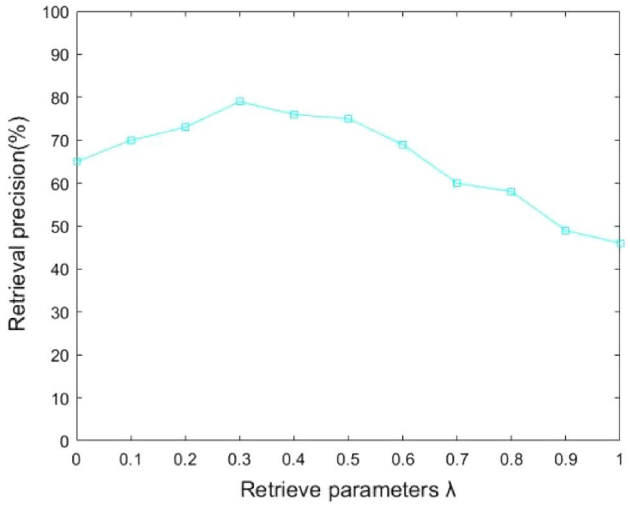


Figure 2. The relationship between parameter λ and retrieval precision.

Algorithm	Data source	PR/%	RE/%	TI/s
YD	Company1	79	86	5
YZ	Company1	75	90	3
DQ	Company1	66	58	12
GS	Company1	42	19	7

Table 1. Accuracy comparison among four algorithms.

Procedure 7: Obtain the comprehensive search vector $\mathbf{Z}_k$	Input: Input utility value retrieval vector in HDFS Output: Comprehensive search vector $\mathbf{Z}_k$

Algorithm 17: Map part

Step 1: Input $\mathbf{T}_k = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_k]$ from HDFS.

Step 2: Combining SimHash similarity and data utility values to calculate the comprehensive score of grid operation data ProRanki, $P_i = (1 \le \mathrm{i} \le k)$.

$$\mathrm{ProRank}_i = \lambda \mathrm{SimHash}(\mathbf{V}_k) + (1-\lambda)U_i, (0 \le \lambda \le 1) \quad (12)$$

From the equation, the system only performs the improved SimHash-based information retrieval when $\lambda = 1$;The system does not perform a comparison mechanism on the retrieved grid operation data. When $\lambda = 0$, the system only compares grid operation data; The system does not consider the retrieved grid operation data to match the SimHash of the retrieved data.

Step 3: Output $\langle \mathrm{key} = \mathrm{ID}_i, \mathrm{value} = \mathrm{ProPank}_i \rangle$.

Algorithm 18: Reduce part

Step 1: Input Map results $\langle \mathrm{key} = \mathrm{ID}_i, \mathrm{value} = \mathrm{ProPank}_i \rangle$.

Step 2: Merge calculation with key as $\mathrm{ProPank}_i$.

Step 3: Output $\langle \mathrm{key} = \mathrm{ProRank}_i, \mathrm{value} = (\mathrm{ID}_1, \mathrm{ID}_2, \ldots) \rangle$

In the output, the comprehensive search vector $\mathbf{Z}_k$ is clustered together. According to ID, the sorting is done sequentially based on the comprehensive score and stored in HDFS and $\mathbf{Z}_k = [\mathbf{z}_1 \, \mathbf{z}_2 \quad \cdots \quad \mathbf{z}_k]$. During the execution of the algorithm, the system retrieves and merges at the same time to improve the efficiency of retrieval. Process 1 to process 3 represents the process of retrieving records. Process 4 and 5 represent the retrieval similarity calculation. Process 6 and 7 are retrieval similarity multi-attribute decision and comprehensive score

Algorithm	Data source	Data size	PR/%	RE/%	TI/s
SimHash	Company1	200	38	85	5
MR-ST	Company1	200	96	92	3

Table 2. Detection results of MR-ST and SimHash algorithm.

calculation. The computational complexity of the algorithm MR-ST is the sum of the complexity of all processes. It can be expressed as

$$O(|i|(3|n| + |j| + |r|)) + O(|i|(4|b| + |r|(1 - \frac{3}{b})) + O(|i|(|k| + |i|))) \tag{13}$$

Experiments and results analysis

In this section, the accuracy and completeness of the MR-ST algorithm are verified, and its parallelism and retrieval efficiency are examined. The experimental setup has 16 nodes as follows: 1 management node, 1 IO node, 14 computing nodes. This experiment builds the cluster experiment environment in Hadoop platform. The system is built with Rocky Linux for computing clusters, configured with 11th Gen Intel(R) Core(TM) i5-1135G7@2.40 GHz 2.42 GHz, 16 GB RAM, 500 GB hard disk, and Hadoop-3.3.0. The system uses NLPIR PARSER Chinese word separation tool. The tool can achieve 96.95% accuracy for single machine word splitting and 982 KB/s speed for word splitting. The experiment used the threshold, similarity length as the similarity calculation criteria in the reference[7]. When two vectors' Hemming distances less than or equal to 3, two vectors are defined to similar. They are called similar vectors and similarity length use 64 bits. For the preprocessing of experimental data is the method in the reference[8]. The operational data of the 3rd quarter of 2021 of a power company in China Southern Power Grid is used as the experimental data for this experiment.

The main indicators to determine the performance of grid operation data retrieval are the search precision rate (PR) and the search recall rate (RE) of the search query. The formulae for calculating the precision and recall of grid operation data are as follows

$$PR = \frac{\sum\limits_{Satisfy} vector}{\sum\limits_{All} vector} \times 100\% \tag{14}$$

where, $\sum\limits_{All} vector$ is the total number of grid operation data files retrieved during the search process; $\sum\limits_{Satisfy} vector$ is the number of all retrieved grid operation data files that match the retrieval requirements.

$$RE = \frac{\sum\limits_{Retrieve} vector}{\sum\limits_{System} vector} \times 100\% \tag{15}$$

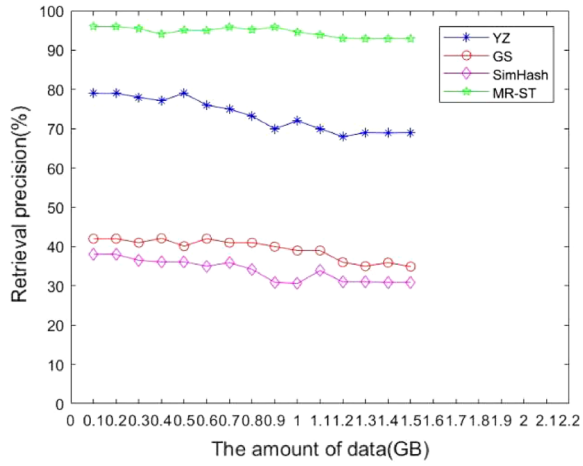
where, $\sum\limits_{Retrieve} vector$ refers to the number of relevant grid operation data files retrieved; $\sum\limits_{System} vector$ indicates the number of relevant grid operation data files in the system.

The amount of data retrieved for all grid operations that meet the retrieval requirements can be preset to a threshold value according to Eq. (12). The higher the accuracy rate calculated by the algorithm, the better the performance of grid operation data retrieval. Obviously, the accuracy of the grid operation data and the related ranking of the retrieved results are directly influenced by the value of λ. After analyzing the experimental results, the relationship between λ and the search accuracy is shown in Fig. 2. As can be seen from Fig. 2, when the value of λ is equal to 0.3, the impact on the retrieval performance of grid operation data is greater, and the retrieval accuracy rate is optimal. Therefore, the utility value calculation based on the TOPSIS method has the greatest impact on the grid operation data retrieval when the value of λ is equal to 0.3.

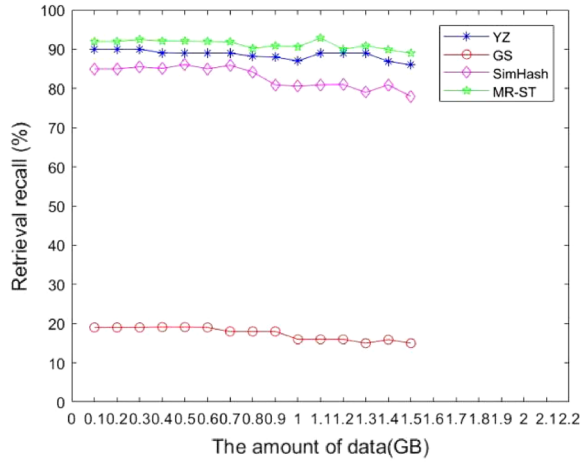
Precision, recall and efficiency analysis of the algorithm. To evaluate the precision, recall and efficiency of MR-ST algorithm, the retrieval results of this paper were compared with those of the four retrieval algorithms involved in the reference[14] and the reference[15]. The compared algorithms are as follows

1. YD is an intelligent commodity information retrieval based on semantic similarity and multi-attribute decision method[14]
2. YZ is an information and knowledge organization intelligent retrieval method based on metadata[15]
3. DQ is multilevel index-driven place name information retrieval method[16]
4. GS is keywords-based temporal information retrieval method over relational databases[17]

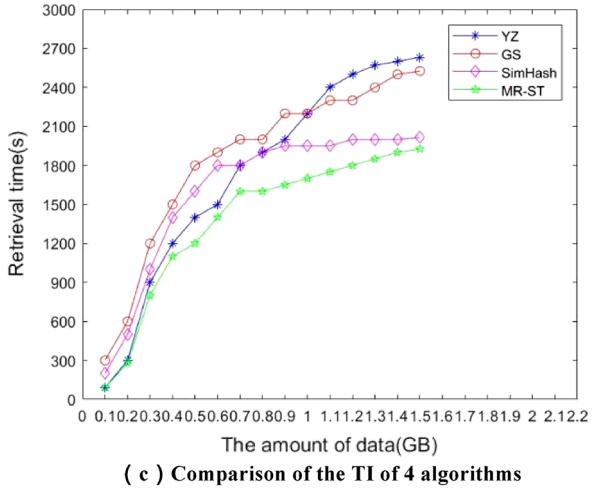
The evaluation standard of the accuracy of similarity search can be measured by PR, RE and search time (TI). The search content is set to be fixed, each automatic search is set to be performed at 0.2 s intervals, and the



(a) Comparison of the PR of 4 algorithms



(b) Comparison of the RE of 4 algorithms



(c) Comparison of the TI of 4 algorithms

Figure 3. Comparison of retrieval performance of several algorithms.

number of iterations is performed 6 times. The PR, RE, TI of the above methods were recorded separately, and the comparison of the results is shown in Table 1.

Similarly, the accuracy of the MR-ST algorithm is verified using a small dataset and the results are compared with those of the SimHash-based retrieval algorithm. The system sets six Map and Reduce processes as experimental tasks; the experimental data contains 12 attributes, the size of the dataset is 200 KB (2500 items) and 6000 KB (25,660 items) respectively, and the detection results of the algorithm are shown in Table 2.

Algorithm	Data source	Data sizes/GB	PR/%	RE/%
SimHash	Company1	2.048	36	81
		20.48	35	83
	Company2	20.48	32	85
		204.8	30	86
MR-ST	Company1	2.048	96	91
		20.48	96	91
	Company2	20.48	95	92
		204.8	95	92

Table 3. The impact of the amount of data on the algorithms.

Data sizes/MB	Runtime/s
10	156
30	312
60	502
100	935
300	1203
500	1355

Table 4. The relationship between runtime and data size.

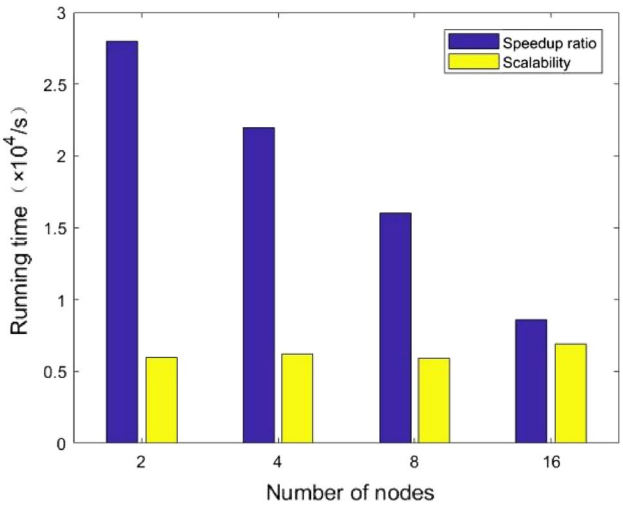


Figure 4. Speedup and scalability of each number of nodes.

The MR-ST algorithm obtained a PR value of 96%, an RE value of 92%, and a TI value of 3/s, so the MR-ST algorithm has good retrieval performance. In addition, the MR-ST algorithm retrieves about 60% more PR, 7.6% more RE, and 40% more TI than the traditional SimHash algorithm. MR-ST algorithm has more accurate and efficient retrieval performance.

From Tables 1 and 2, it can be seen that the SimHash algorithm has worse retrieval performance than the MR-ST algorithm, and the MR-ST algorithm has the best retrieval performance. Moreover, the PR, RE, and TI values of MR-ST algorithm are higher than those of other algorithms, so MR-ST algorithm has the optimal retrieval performance. In addition, among the four algorithms in Table 1, the YZ algorithm has the best balance of PR and retrieval time, and its performance is optimal. The GS algorithm has the lowest PR and RE, and also takes relatively long to search, and has the worst balance, so its search performance is the worst. To verify the retrieval performance of the proposed MR-ST algorithm in this paper, a larger dataset is used and compared with the traditional SimHash algorithm, the YZ algorithm with the strongest retrieval performance and the GS algorithm with the worst retrieval performance in Table 1. In the experiment, 12 Map and Reduce task processes are set, and the experimental data comes from the operational data of a power company in China Southern

Power Grid. The data size is 1.5 GB, and each record contains 11 attributes. The obtained experimental results are shown in Fig. 3.

From Fig. 3, it is clear that the MR-ST and YZ algorithms have the highest execution efficiency under the same conditions. However, the PR and RE of the MR-ST algorithm are higher than those of the YZ algorithm. Because the MR-ST algorithm directly outputs the similarity retrieval vector during the execution and performs multi-attribute decision making and comprehensive scoring on it. The algorithm simultaneously merges and retrieves similarities. Therefore, MR-ST has short running time and highest efficiency. In addition, compared with the SimHash algorithm, the MR-ST algorithm improves the retrieval efficiency by about 26%, has the highest PR, RE, and TI values, and has the best retrieval performance.

Influence of parameters on the algorithm. *Relationship between data size and PR, RE.* The experiments were run for 8 MapReduce task processes for data sizes of 2.048 GB, 20.48 GB, and 204.8 GB to verify the effect of data sizes on the PR and RE of MR-ST and SimHash algorithms, and the obtained experimental results are shown in Table 3.

As can be seen from Table 3, data sizes do not affect PR and RE values, but data source does. Because of the differences in the data structure of different data sources, the retrieval performance of the MR-ST algorithm will show a small fluctuation (PR value floats within 2%, RE value floats within 2%). Even so, it is enough to meet the real-time requirement of similarity retrieval of massive grid operation data. Meanwhile, the PR value obtained from SimHash algorithm retrieval fluctuates within 6.6% and RE value fluctuates within 2.5%. Hence, the impact of data size on MR-ST algorithm is smaller and more stable among different data sources, which is more suitable for intelligent retrieval of rapidly growing huge amount of grid operation data.

The relationship between data size and algorithm efficiency. The experimental data used for the 8 MapReduce task processes running in the experiment contains 12 attributes. Experimental data sizes are as follows: 10, 30, 60, 100, 300, 500 MB. The results of the retrieval efficiency of the proposed MR-ST algorithm affected by the size of retrieved data are shown in Table 4.

From Table 4, it is observed that the larger the data size, the higher the retrieval efficiency of the MR-ST algorithm. When the data size is small, the search time will be significantly longer. But when the data belongs to a large size, the rising trend of retrieval time will slow down. The reason is that the parallelism of MR-ST algorithm can be fully utilized when the data size is large, and the retrieval efficiency can be maximized. MR-ST algorithm can still maintain a certain retrieval efficiency when all parallel tasks are started. The retrieval efficiency of the MR-ST algorithm proposed in this paper is positively correlated with the scale of data and is suitable for intelligent retrieval of massive grid operation data.

Speedup and scalability. The number of nodes is a measure of the system speedup and represents the number of parallel execution processes in the system[6]. speedup refers to the ratio of the time consumed by the same task running in a parallel processor system. It is used to measure the performance and effectiveness of parallel algorithms when the data size is fixed and the number of nodes is increasing. The ideal speedup varies linearly. Scalability refers to improving performance and availability at a larger scale. It indicates the performance of the parallel algorithm when both the number of nodes and the size of the data grow proportionally[7]. For the evaluation of the algorithm speedup, the experiment uses 20 GB of grid operation data volume with grid data size of 2.5, 5, 10, and 20 GB. For the evaluation of the algorithm scalability, experiments were conducted using the number of nodes of 2, 4, 8, 16. The obtained results are shown in Fig. 4.

The proposed MR-ST algorithm has a high speedup ratio and good scalability. However, because of the computer communication overhead and other problems still exist, the acceleration ratio cannot reach the ideal state. The main reason for this is the consumption of computer software and hardware resources, etc. The running time consumption of the algorithm rises slightly when the number of nodes reaches 16. However, the overall execution efficiency of MR-ST algorithm is basically the same, so it has good scalability.

Conclusion

(1) The MR-ST algorithm has a precision rate (PR) > 95% and a recall rate (RE) > 91%. The search precision rate (PR), search recall rate (RE) and search time (TI) of the algorithm are negatively correlated. The algorithm is efficient and stable, and the data size does not affect its PR and RE values, with good retrieval performance and more reliable applicability.

(2) The MR-ST algorithm executes efficiently and is positively correlated with data size. The algorithm can perform similarity retrieval with multi-attribute decision and composite score, and can directly output the top similarity vector.

(3) MR-ST algorithm has excellent speedup ratio and scalability, suitable for intelligent retrieval of massive grid operation data. Future research will focus on optimization algorithms for massive grid operation data. For example, the impact of a larger number of nodes on the algorithm's retrieval performance. The optimization algorithm will be better applied to the huge amount of grid operation data and can provide data support to enterprises.

(4) The proposed approach is applicable to the intelligent retrieval of grid operation data, and it can effectively improve the efficiency of data acquisition and mining of key business analysis and decision making for each power company in real-time.and specifically that, the "Intelligent Retrieval Method for Power Grid Operation Data Based on Improved SimHash and Multi-Attribute Decision Making" has been applied in the scientific research project named "Research on Vulnerability Defense Technology of Power Monitor-

ing System Based on Interdependent Network (047000KK52210031)", which is used for the optimization design of the "data intelligent retrieval module of the massive abnormal data correlation analysis system". The system module has been applied in the field of power grid engineering and has been recognized by the application unit.

Data availability

The data that support the findings of this study are available from [Power companies in China] but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. Data are however available from the authors upon reasonable request and with permission of [Power companies in China].

Received: 26 June 2022; Accepted: 29 November 2022
Published online: 05 December 2022

References

1. Liu, B. Research on power system operation evaluation and optimization based on power grid operation data set. Master's thesis, Beijing, North China Electric Power University. (2017).
2. Zhang, D. *et al.* Research on development strategy for smart grid big data. *Proc. CSEE.* **35**(01), 2–12 (2015).
3. Song, Y., Zhou, G. & Zhu, Y. Present status and challenges of big data processing in smart grid. *Power Syst. Technol.* **37**(04), 927–935 (2013).
4. Yi, W. *et al.* Clustering of electricity consumption behavior dynamics toward big data applications. *IEEE Trans Smart Grid* **7**(5), 2437–2447 (2016).
5. Zhu, J. Research and application of time-series data retrieval method for smart grid. Master's thesis, Chongqing, Chongqing University of Posts and Telecommunications. (2021).
6. Song, R. *et al.* A similar duplicate record deiection algorithm for big data based on MapReduce. *J. Shanghai Jiaotong Univ.* **52**(02), 214–221 (2018).
7. Manku G S, Jain A & Sarma A D. Detecting Near-Duplicates for Web Crawling. WWW2007; International world wide web conference. Google Inc.; Google Inc.; Stanford University (2007).
8. Ye, X. *et al.* Improved approach to TF-IDF algorithm in text classification. *Comput. Eng. Appl.* **55**(2), 104–109 (2019).
9. Wang, J. & Zhang, J.-S. Comparing several methods of assuring weight vector in synthetical evaluation. *J. Hebei Univ. Technol.* **30**(02), 52–57 (2001).
10. Jin, Z. A method of intelligence key words extraction based on improved TF-IDF. *J. Intell.* **33**(04), 153–155 (2014).
11. Balabanovi, M. & Shoham, Y. Fab: content-based, collaborative recommendation. *Commun. ACM.* **40**(3), 66–72 (1997).
12. Zeng, Z. & Zhang, L. An intelligent shopping system based on multi-attribute decision and collaboration filtering. *Eng. J. Wuhan Univ.* **02**, 136–140 (2008).
13. Qu, Z. *et al.* An attribute reducing method for electric power big data preprocessing based on cloud computing technology. *Autom. Electr. Power Syst.* **38**(08), 67–71 (2014).
14. Zeng, Z. & Zhang, L. An intelligent commodity information retrieval based on semantic similarity and multi-attribute decision method. *Data Anal. Knowl. Discov.* **01**, 22–27 (2010).
15. Wang, D., Zhang, X.-h & Zhao, H.-y. Design of information and knowledge organization intelligent retrieval system based on metadata. *Inf. Sci.* **37**(09), 113–116 (2019).
16. Li, A. Multilevel index-driven place name information retrieval method. *Sci. Surv. Mapp.* **42**(04), 103–107136 (2017).
17. Zhang, X. *et al.* T-STAR: keywords-based temporal information retrieval method over relational databases. *Appl. Res. Comput.* **34**(10), 3051–3056 (2017).

Acknowledgements

The paper is supported by Research on Vulnerability Defense Technology of Power Monitoring System Based on Interdependent Network (047000KK52210031).

Author contributions

S.Z. (First Author and Corresponding Author): conceptualization, methodology, investigation, formal analysis, writing—original draft; X.G.: Conceptualization, funding acquisition, resources, supervision, writing—review and editing; Z.Q.: data curation, visualization, investigation; Z.Z.: resources, supervision, validation; T.Y.: visualization, writing—review and editing.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to S.Z.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.