Search Algorithm Study Based on Big Data Technology in Civil Aircraft Research and Development

Hongxia Cai [1, a], Minshan Ren[2, b]

[1]School of Mechatronic Engineering and Automation, Shanghai Key Laboratory of Intelligent Manufacturing and Robotics, Shanghai University, Shanghai 200070, China.

[2]School of yyy, zzz University, Guangzhou 510000, China. Graduate Institute of Mechatronic Engineering and Automation Shanghai University, Shanghai 200070, China

[a)] hxcai@shu.edu.cn

[b)] 546600130@qq.com

Keywords: Big Data, TF-IDF algorithm, Civil Aircraft

Abstract. Comparing with the search function focusing on massive unstructured process data in current civil aircraft system, the improved search algorithm proposed in this paper mainly makes the following optimization:At first, the data is preprocessed before segmentation to enhance the precision and speed;Secondly, considering that keywords in different places weight differently, a concept of key coefficient is introduced based on the traditional TF-IDF algorithm, and for a better representativeness of the extracted words, threshold in the weight of keywords is set as extracted words filters; Then, the improved TF-IDF algorithm is based on the big data programming model MapReduce to improve the computational efficiency. Final, including the feedback of individual query, the improved sorting algorithm can provide a better query result to meet users' demand.

Introduction

The civil aircraft research and development (R&D) is a typical industry of high-end equipment. It is a rather complicate systematic project from making sure demand to putting into use, especially the production and manufacturing, in which the assembly is wholly guided by AO (Assembly Outline). Drawing up the AO needs the knowledge related to original AO, such as process management and so on. The corpus in this paper is mainly the unstructured process data of AO. And the aim is to build a vertical search engine of process in civil aircraft R&D [1] .

Actually, during the development and manufacturing in civil aircraft corporation, every type of aircraft has a large number of AO, and at least several millions of process data, which cannot be utilized effectively for being dispersed throughout all application and server within the corporation. However, the big data searching technology [2] and enterprise search engine [3] has changed this situation. This paper propose to make a unified storage of the unstructured process data dispersed within the corporation through distributed storage technology, provide a unified information search interface for corporations by big data parallel processing techniques, and introduce personalized techniques into enterprise search engine for civil aircraft R&D. To be more detailed, in accordance with users' different degree of attention to different information, the query result of higher attention will be ranked in the front of the result list while the one with shorter attention will be ranked behind. Thus the query result of enterprise search engine can meet the personalized demands better. Until now the several famous sorting algorithms are the term frequency/ position weighing algorithm[4], Direct Hit algorithm[5] and PageRank algorithm[6] [7].

Combining the shortcomings and merits of sorting algorithms above with the process data features in civil aircraft R&D, this paper puts forward an improved algorithm based on TF-IDF (term frequency-inverse document frequency) [8]. At the same time, the improved sorting algorithm is based on the dispersed storage and parallel computing of big data frame work Hadoop[9] .

Search Corpus Building

This paper chooses all process data in AO as the main source of corpus building for three reasons. First, the whole assembly phase in civil aircraft R&D should be guided by process. Therefore the reasonable and effective process can greatly shorten the R&D cycle and lower the cost. Second, in current civil aircraft R&D system, the process is mainly stored in the value of database table by CLOB data type as a piece of semi-structured XML data. As a matter of fact, it is very difficult for current business system to do fuzzy matching search about the unstructured data, unless the CLOB data type is converted to character type. Nonetheless, the storable length of each value is limited after changing, which obviously does not meet the requirements. Moreover, this method needs to traverse every long text in the database in fuzzy matching, and thus will be rather inefficient when the amount of data is rather large. Meanwhile, the pieces of matching data cannot be ranked by relevance. Neither can the quick fuzzy matching be realized because some processes are saved as PDF, Words in the servers directly. Third, users cannot achieve a unified query since a large amount of process data is dispersed in different business servers such as MES and PDM. The unified query can be achieved by associated query through tables or writing several SQL statements to query different tables, and then connect the tables manually. These approaches are onerous when the data is voluminous.

The paper will make a unified extraction of the voluminous process data dispersed in business servers by tools like ETL. Take the name of AO as the title, filled by all process in the AO, stored as key/value in database, structures as Table 1, every document in searching corpus is built in this way.

TABLE 1. The process structure in unified storage

Document title	Document content
The lubrication of hydraulic lines behind fairing in the right wing body of iron bird	According to engineering drawing 185A4101, FL6, apply BMS3-24 grease to bearing at the back of wing body for lubrication and temporary protection. Check whether the lubrication is completed. And write all information required on a qualified label. …
…	…

Improved Search Sorting Algorithm

TF-IDF Algorithm

TF-IDF algorithm is a weighting technology usually used in information query and data mining. It is a statistic method to evaluate the level of importance of a word to a document set or one document in database. Taking into account of both the term frequency and the number of documents containing this word, this method can ensure the limited amount and distinction of extracted feature word, and is suitable to show the connection between a word and a document.

Calculation Method of User Interest

Clicks and time on page[10] are two significant characteristics of user interest. According to the thoughts of Direct Hit sorting algorithm, this paper appropriately takes this two factors as rank weighting, in order to link user interest with final sorting.

Many references[11] introduce how to transform clicks and time on page into user interest. Yet these traditional means cannot really reflect user interest. Combined with measures mentioned by the two references [10][12], this paper proposes a new calculation method. With this method, the user interest has a positive correlation with clicks and time on page, and a monotonic distribution between 0 and 1, providing convenience for the subsequent calculation. Here is the calculation formula:

$$I_i = \frac{2}{\pi}\arctan\left(\frac{\text{the number of times to click i for users who query Q}}{\text{the average clicks for users who query Q}} + \frac{t_i}{\text{averagetime}}\right) \quad (1)$$

In this formula, I_i denotes the degree of user interest in the ith website; t_i represents browsing time on the ith website; averagetime means the average time for browsing; and π is the ratio of circumference.

Data Preprocessing

Because the isomerism of process data format and many useless information symbol in the data, this paper will preprocess data before segmentation to enhance the precision and speed, which comprises the following steps:

Use Tika[13] to transform all isomerized documents into unified structured ones (such as TXT documents);

Divide the processed document set into string set by punctuation. Meanwhile, determine whether there is a tag; Remove the tag if it exist and save the string according to the document it belongs to; repeat this loop;

Remove stop words, such as articles, auxiliary words and modal particles, in the processed string set in accordance with stop word dictionary; repeat this loop;

Improved Sorting Algorithm Based on TF - IDF Algorithm

In the corpus built in this paper, some meaningless words, such as of, this and other functional words, with a high frequency in a text, might have larger TF-IDF values calculated by traditional TF-IDF algorithm and be extracted as keywords. According to the structure features of process documents in the corpus, it is easy to know that just a title can summarize the main idea of whole document, which means a keyword in title is much more important than the one in content. However, TF-IDF algorithm cannot take this factor into account.

On the basis of analysis above, this paper also aims at improvements on TF-IDF algorithm. First of all, taking into account the content continuity of process document, this paper does not consider paragraph position in document, but takes the document as a whole. And then the title of document is considered as the keyword in title is always significant. A concept of a key coefficient in introduced in this paper to improve the TF-IDF algorithm formula. The strategies are as follows:

As to key coefficient, assuming that a specific document T comprises part K1 and part K2, where K1 is a key coefficient of title and K2 is another one of content, it is apparent that if a keyword appears in title, its importance is self-evident, and thus a larger value is given to K1. Therefore, the importance of keyword can be determined by its location in the document. In accordance with the analysis above and the structure features of process document, this paper determines the weight distribution through empirical data and proposes the improved TF-IDF algorithm formula below:

$$\mathrm{TF}-\mathrm{IDF}=(0.5(\frac{TF_X}{TF_{max}})+0.8(\frac{K_n}{K_1}))\times\log\frac{|D|}{1+|\{j:t_i\in d_j\}|} \quad (2)$$

In this formula, TFx means the term frequency of keyword x in document, while TFmax represents the maximum term frequency of keyword in a particular document. Kn indicates key coefficient of some place (n is 1 or 2); K1 is key coefficient of title, and K2 is of document content.

Analyzing sorting methods of common search engine, especially with the reference of Google's sorting factor, this paper selects and analyses some factors shown in Table 2 below, which are certainly not all factors affecting sorting result, but are rather essential in the view of relationship between words and documents and user behavior.

TABLE 2. Important factors affecting sorting

Factors that affect the sorting of search results	Factors
Relationship between words and documents	Frequency of keyword in documents
	Location of keyword in documents
User behavior	Clicks of documents
	Browsing time on documents

The relationship between words and documents

First, the higher frequency the keyword has in document, the easier this document to be searched, and the higher it is placed on the result list. Considering about the location of keywords in documents, this paper focuses on two relationships between title and content, and the score of relationship between words and documents is calculated as formula (2). Since keywords in title are much more important, K1=0.6, K2=0.4 are also set in this paper.

User behavior

Clicks of document: The more the clicks of document are, the more popular to keyword of search

the document is. And the document is placed high on the result list.

Browsing time on document: The longer the browsing time is, the more valuable information the document has. And the document is placed high on the result list.

The final score of user behavior is calculated as formula (1).

Based on the analysis above, the improved algorithm takes into consideration the term frequency and location of words in document and user behavior feedback. Here is the improved algorithm formula:

$$\text{score} = \text{k1} \times \text{score1} + \text{k2} \times \text{score2} \quad (3)$$

score: comprehensive score of queried document;

score1: score of word and document relationship;

score2: score of user behavior feedback;

K1, K2 are weighting coefficient.

The final sorting result depends on the comprehensive score of queried document. Fully considering about the several important factors affecting search result in actual business system, this algorithm efficiently overcomes the shortcomings of basic TF-IDF sorting algorithm.

Implementation of Improved TF-IDF Algorithm under the Hadoop Framework

Distributed Parallel Computing Platform of Hadoop

The Hadoop is composed of Hadoop Distributed File System (HDFS) and MapReduce Programming Model. On the one hand, the HDFS is mainly tasked with data storage on each node and fulfills the high throughput of data reading. On the other hand, the MapReduce Programming Model consists of two functions, Map and Reduce. This two user defined functions transform an input key-value pair into another one or a set of key-value pairs in accordance with certain mapping rules.

Implementation of Improved TF-IDF Algorithm under the Mapreduce Programming Model

The core idea of Mapreduce programming model is to split the task and then run it in parallel. Shown as the improved TF-IDF algorithm formula (2), it is definitely suitable for distributed computing. The term frequency (TF) of keywords only depends on the total number of keywords and the number of times a keyword occurs in a document. Therefore, the calculation can be sped up by splitting the data and parallel counting the TF of keywords in each document. After that, calculating the weight of keywords TF-IDF depends on number of documents containing this keyword (as the total number of documents is a constant) and keyword location in the document, which means the calculation of TF-IDF value can be improved through parallel computing. This paper designs that TF-IDF value is calculated through Map and Reduce functions three times.

Counting the Number of Occurrence and Location of Keywords in Each Document

After sharding, the original data is passed to the Map function. In this function, the keyword is recognized through regular expressions, written to intermediate results in the form of key/value pair <word#documentName, 1>, and then passed to the Reduce function, in which both number and location of keywords are counted. Subsequently, the results are outputted to tempFile1, a temporary file that will be an input of MapReduce in the next step. The key of these results is <word#documentName>, and the value is <n(position)>, with n referring to the number of times the keyword occurs in the document of documentName, and position indicating the keyword location in the document, which mainly contains two types, title and body of documents. The functions are designed as follows:

Map ():

Input: <documentLineNumber, contents>

Output: <word#documentName, 1>

Reduce ():

Input: <word#documentName, 1>

Output: << word#documentName >, n(position)>

This step is to count the number of times that keyword occurs and its location in documents.

Counting the TF of Keyword in Documents

Take the temporary file tempFile1 in the previous step as the input of the Map function this time, in which the key/value pairs (using <word#documentName> as the key, and <n(position)> as the value) are rearranged for the calculation convenience of Reduce function in the next step. In this function, to count the total number of keywords only needs to add up the keyword numbers in every document. Then saving the output as another temporary file tempFile2, the results will be an input of MapReduce for the calculation of TF-IDF value in the next step. The functions are designed as follows:

Map ():

Input: << word#documentName> , n(position)>

Output: < documentName, word=n>

Reduce ():

Input: <documentName, word=n>

Output: <<word#documentName, <n/N>>

In this step, the TF of keyword in each document is counted.

Calculating the Improved TF-IDF Value

Input tempFile2 of the previous step into the Map function. In this function, rearrange the key/value pair (using keywords as the key and <documentName=n/N> as the value) for the convenience of calculating d, the number of times the keyword occurs in the document set, and D, the number of whole document set. According to formula (2), the algorithm that calculates the improved TF-IDF value is designed as follows:

Map ():

Input: << word#documentName> , n/N>

Output:<word, documentName=n/N>

Reduce ():

Input: <word, documentName=n/N>

Output: <<word#documentName, improved TF-IDF value >

This step can calculate the improved TF-IDF value of keywords corresponded to each document in the document set. To conclude, the entire flowchart is shown as Fig. 1.

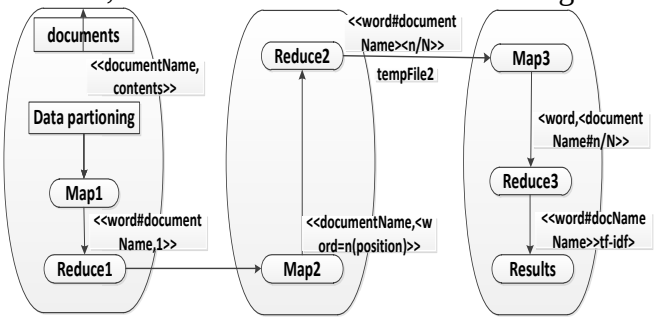

FIGURE 1. Flowchart of improved TF-IDF algorithm under the MapReduce programming model

Example Verification

The experiment in this paper is to test and evaluate the most common parameters in search engine: precision, recall rate and system response time. Its evaluation method is to compare the automatic extracted results with manual labeling ones.

Precision formula:

$$P=\frac{|A\cap H|}{|A|} \tag{4}$$

Recall rate formula:

$$R=\frac{|A\cap H|}{|H|} \tag{5}$$

In these formulas, P indicates an ability of system to extract keywords, while R denotes an ability to find; A is automatic extracted keyword set, while H is manual labeling one; |A∩H| means the same keywords of both, while |A| and |H| respectively represent the number of words that are unique to each other.

After referring to many references, many times of taking values and experiments in different intervals, k1 = 0.48 and k2 = 0.52 are set in the formula (3). Using python language to achieve the algorithm.At the same time, the distributed environment consists of five nodes,one Master and four Slaves.Operating system for Ubuntu12.04,memory 4GB.The configuration of the cluster environment is as follows:the configuration capacity of HDFS is 1.72TB,and the HDFS configuration of five nodes is shown in Table 3.

TABLE 3. Configuration capacity of HDFS

	The capacity of HDFS/GB	Number of Map tasks	Number of Reduce tasks
Master	318.75	0	0
Slave1	302.75	2	2
Slave2	302.75	2	2
Slave3	302.75	2	2
Slave4	302.75	2	2

In order to be close to the application environment of practical system, there are one hundred thousand pieces of data, which corresponds to one hundred thousand million process documents in the corpus.

Primarily, input 20 popular keywords selected from the search corpus built in this paper as a test, and then record their precision, recall rate and response time. For precision, only top 100 query results are compared to calculate the precision. As to recall rate, the amount of tested data is too large to be manually counted the number of actual reasonable results corresponded to a certain keyword. And thus this paper takes as estimating method as below: take the number of elements in the reasonable result union of two algorithms as total reasonable number, dividing the total number by the number of reasonable data of one algorithm as the recall rate of this algorithm. Assuming that there are S0 approximate data records detected by an algorithm and the amount of approximate data records collected by the two algorithms in the data set is S, then the recall rate of this algorithm is R = S0 / S.

In this experiment, the improved search sorting algorithm discussed above is compared with traditional TF-IDF algorithm. The former one is based on the MapReduce programming model and runs on the distributed environment while the latter one runs in a stand-alone environment. Fig. 2 shows the comparative results of the two algorithms in response time.

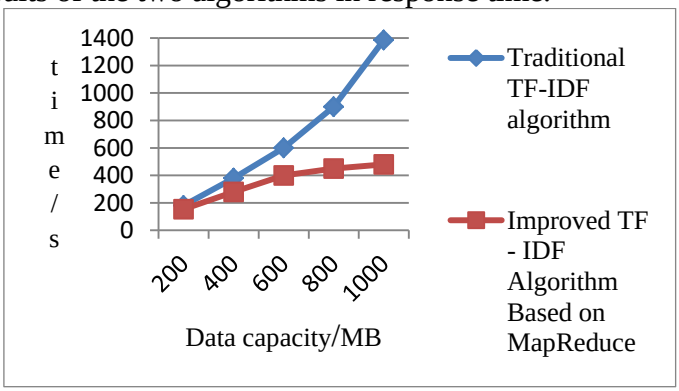

FIGURE 2. Response The comparative results in response time

It can be seen from Fig. 2, when the amount of data is small, the performance gap between improved TF-IDF algorithm based on MapReduce and traditional TF-IDF algorithm is not obvious. Since Hadoop is fragmented on data and has a default block size of 64MB, the speed of operation is reduced when there are many small data files, so the superiority of Hadoop for the small data set is not obvious. However, as the data set increases, the time required for traditional algorithms increases dramatically, and the time required for the TF-IDF algorithm of the Hadoop framework is linearly

increased, showing a certain degree of superiority. Fig. 3 shows the comparative results of the two algorithms in precision and recall rate.

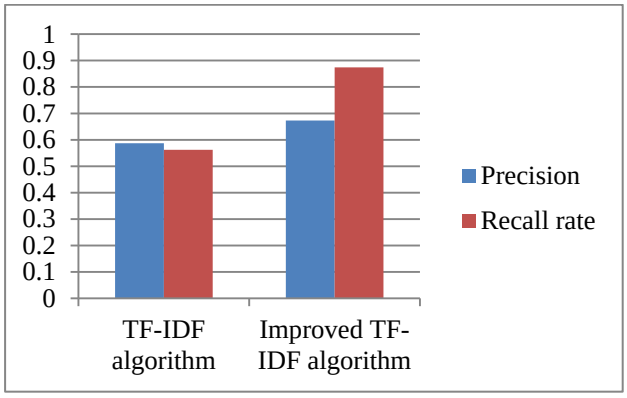

FIGURE 3. The comparative results in precision and recall rate

It can be seen from Fig. 3, compared with the traditional TF-IDF algorithm, the precision and recall rate of improved TF-IDF algorithm are visibly improved. The precision of 72.6%, higher than that of traditional TF-IDF algorithm, shows the document placed high has higher term frequency and more important location for keywords. And the recall rate, 87.4%, also higher than that of original one, indicates that the document placed high has more clicks and longer browsing time.

Conclusions

In this paper, the traditional TF-IDF algorithm is improved and is achieved by MapReduce programming model under the Hadoop framework, while the improved search algorithm takes the users' behavior into account.By the contrast experiments between the improved TF-IDF algorithm based on MapReduce and the traditional TF-IDF algorithm for the response time,preision and recall rate, it is verified that the response time of the improved TF-IDF algorithm based on MapReduce is obviously shorter than the traditional TF-IDF algorithm When the amount of data is large enough, while the precision and recall rate were significantly improved. Meanwhile, the definition of precision and recall rate in this paper is reasonable as well,because some parameters cannot be counted precisely in the practical vertical search engine with massive data.In conclusion, the process search engine built in this paper can promote the search effectiveness, results ranked on the front page can meet the needs of users better, which is friendly for users to query process related knowledge better and faster.

Acknowledgment

This work was supported by the project: Shanghai Municipal Comission of Economy and Informatization, big data application demonstration project construction to enhance aircraft development capacity, project code: 201502016.It was supported by Shanghai Key Laboratory of Intelligent Manufacturing and Robotics.The authors are grateful for the financial support and also would like to thank the anonymous reviewers and the editor for their comments and suggestions.

References

[1] Ye Wang. Research on Several Problems of Vertical Search Engine. Fudan University,2011.

[2] Shichang Hu. Research of the Distributed Search Model based on Map/Reduce [D]. Xiamen University,2014.

[3] Chunyan Li. Research and Implementation of Vertical Search Engine in Enterprise Information. China University of Geosciences (Beijing),2010.

[4] Atsushi Masumura. Effect of relationships between words on Japanese information retrieval. ACM Transactions on Asian Language Information Processing. 2006:264-289.

[5] M.M.Sufyan Beg.A subject measure of web search quality.Information Science-Informatics and Computer Science:An International Journal.2005:365-381.

[6] Jun Cao. Technology Analysis on PageRank of Google. Journal of Intelligence, 2002,(10):15-18.

[7] Maththew Richardson.Beyond PageRank: machine learning for static ranking.International Word Wide Web Conference.2006:707-715.

[8] Yonghe Lu, Yanfeng Li. Improvement of Text Feature Weighting Method Based On TF-IDF Algorithm. Library and Information Service, 2013,(03):90-95.

[9] Li Bin. Improve of TF-IDF algorithm based on Hadoop[J]. Microcomputer &. Its Applications, 2012,31(7):14-16.(in Chinese)

[10] Xingwu He, Zhengyu Zhu, Xin Deng. Application of Browsing Time in User Profile [J]. Information Science, 2008,26(1):97-100.

[11] Schwab L., Pohl W. and Koychev I. 2000. Learning to Recommend from Positive Evidence. In Proceedings of the International Conference on Intelligent User Interfac. New York: ACM Press, 241-247.

[12] Yiqun Liu, Rongwei Cen, Min Zhang, liyun Ru, Shaoping Ma. Automatic Search Engine Performance Evaluation Based on User Behavior Analysis. Journal of Software, 2008,(11):3023-3032.

[13] Xuren Wang, Qiuhui Zheng, Famei He, Na Li, Yanli Wang. Research and implementation of desktop search engine based on Tika and Lucene. Computer Engineering and Design, 2014,(01):310-314.