Research on Search Method Based on Data Segmentation of Related Attributes

Zhong-wen QIAN¹, Jian-son ZHANG¹, Xiang WU²,* and Xiao-ming JU²

¹StateGrid Zhejiang Electric Power Company, Information and Communication Branch, Hangzhou, China
²East China Normal University, School Computer, Science and Software Engineering, Shanghai, China

*Corresponding author

Keywords: Data segmentation, Apriori, Word vector, Index.

Abstract. Aiming at the problem of safe protection and effective searching of texts in cloud environment, as well as improving the validity of cloud storage search, this paper proposes a text search method based on data segmentation of text attribute relationship. The index construction of the split text search method is carried out by firstly using the improved Apriori algorithm to obtain the associated word set of the search text to be uploaded, and then utilizing the associated word sets to split the text. Combining the text associated attribute word set and the text word vector space model, the keyword sets which are required for constructing the text index is selected. Experimental analysis shows that the search method constructed in this paper can achieve effective text search and restoration, and is suitable for text storage and search in complex cloud environments.

Introduction

Cloud computing[1] combines related technologies such as networks and virtualization and provide services to users. The cost of information storage in cloud environments is low, and the space of storage can also be allocated according to the needs of users [2]. However, cloud storage is currently faced with the following problems: the unstructured characteristics of data stored in the cloud, and the lack of machine-understandable semantics, which can result in imprecise research of the required text when searching for cloud storage. The accuracy of cloud storage search is mainly reflected in extraction of text feature information. Text feature selection algorithms include text frequency[3], information gain [4], CHI(chisquare) [5] etc., but which have their own limitations, such as the text frequency algorithm, taking into account the frequency information of the text vocabulary, but did not consider text distribution and associated information for vocabulary. Extracting the characteristic information of the important description text in the text to build an effective index is the core of the cloud storage search.

At present, many scholars are doing tireless research in this area. In 2013, Kamara[6] et al introduced the red-black tree structure as an index structure based on text features and used symmetric dynamic searchable encryption to support parallel retrieval of multiple processors. In the same year, Kamara proposed a parallel homomorphic encryption scheme and studied various MapReduce operations including textual keyword search under the MapReduce parallel computing model. In 2013, Wu Qi[7] made improvements to Yang keyword search solutions, based on which the user must give full keywords. In 2014, Chen Hefeng [8] given a ciphertext-based Chinese keyword fuzzy search scheme to study the improvement of existing search methods based on the similarity of Chinese Pinyin characters. In 2016, Su Xiaowei[9] proposed constructing a parallel search scheme based on text feature information, which reduced the index construction time and improved the performance of searching large files.

This paper proposes a retrieval method based on association attribute data segmentation. It takes the information of related words, frequency distribution, and paragraph distribution of texts into
account to select document ciphertext entries. Under the hybrid cloud environment, a segmented
document search method based on association attributes is constructed.

**Research on Searching Method of Segmentation Text Based on Associated Attributes**

**Text Segmentation**

There are various types of file splitting methods. Most of the splitting operations are based on random
number splitting. The principle is based on the input stream to read text, output to the specified file
according to a certain output stream. The flow chart is shown in Figure 1.

![Flow Chart](image)

Figure 1. Split flow chart.

The random segmentation method uses a random function to segment the text. The algorithm
begins to set a range of random numbers, generates a random number sequence, sorts by size, extracts
bytes, sets the extracted bytes into a set of data blocks, and uses the remaining blocks as large data.
The method does not combine the information entropy of the text, ignoring the segmentation to
combine the text information.

**Text Segmentation Based on Associated Attribute Sets**

Concurrency relations between entities are called association[10], and also for the existence of
vocabulary in document. If n(n>=2) words in the vocabulary appear in the paragraph set more than a
certain threshold, It is said that there is a certain relationship between these n words, such as table
tennis and ping pong rackets, toothbrushes and toothpaste. If several items often appear concurrently
in the same large category, there is a correlation between these items.

Defining a text vocabulary $word=\{w|i=1,2,...,n\}$, n express the size of glossary. Defining text paragraph
sets $par=\{p|j=1,2,3,...,m\}$, m represents the number of paragraphs in the text. The number of
paragraphs that contain the term $w_i$ is defined as $pa=\{d|k=1,2,3,...,tf\}$ tfi represents the total number
of paragraphs in the text that contain the word $w_i$. For a text, the word segmentation is processed
through a vocabulary. Regardless of the frequency relationship and sequence relationship that occurs
internally, it can be seen as a word set V about the text. Similarly, for any paragraph d of the text, it is
obtained. vocabulary $v$, $v \subseteq V$, and $v \subseteq V_d$, can we think Vd contain v. The degree of support for the
related word set v is the percentage of the number of text paragraphs that contain the related set of
terms v, and defines the support degree $support(v)$ in (1).

$$support(v) = \frac{sum(v)}{m}$$ (1)

$sum(v)$ represents the number of paragraphs in text that contain vocabulary v
The Apriori [11-12] algorithm shows that in order to find strong association rules, frequent sets must be found first. The Apriori algorithm proposes a downward closure for finding related words (frequent sets for short), generating 1-frequent sets, generating 2-frequent sets from 1-frequent sets, ..., generating (x-1)-frequent sets, and finally finding all frequent word sets.

Improved Apriori algorithm for finding correlation sets and x-frequent sets:

Begin

Input: (t-1)-frequent sets;
Output: t-frequent sets;

Find all the different (t-1)-frequent sets pair sets, two of them feature is the different of the last element, next combine of them into candidate t-frequent set;
for each t-frequent set find its (t-1)-frequent sets
if one is not in the input of (t-1) frequent sets
remove it from candidate;
if support(t-frequent set) < mindown(t)
remove it from candidate;
return candidate t-frequent sets;

End.

The above algorithm generates x-frequent sets by downward closure way. The traditional Apriori algorithm generally uses a fixed support degree. In the above algorithm, the activity support function mindown(t) is selected to adjust the size of the frequent set, and the size of the support formula (2) is defined.

\[
\text{mindown}(x) = \frac{t!}{2^t}
\]  

(2)

The experiment found that by the degree of support gradually decreasing, the number of frequent sets shows an exponential increase. With an index size close to two, these redundant frequent sets will only cause unnecessary index disasters. Therefore, the above constraints will not the necessary part is removed and the frequent set can be better selected.

Now, we can use the generated frequent set to help achieve text segmentation. For a text, generates its frequent set through the above algorithm \( T = \{ t_i | i = 1, 2, ..., t \} \), \( t_i \) corresponds to a set of related x-frequent sets, where \( x \) represents the size of the frequent sets. For a piece of textual content, if you select a word through the paragraph to cut the paragraph, the paragraph will form two parts. Definitions: Once an x-frequent set has been completely segmented a text, approximately cut the text into \( x+1 \) parts, so the important thing is to choose a suitable sets. Using formula (3) help randomly select a pd(T)-frequent set. \([x]\) indicates rounding.

\[
\text{pd}(T) = \left[ \frac{\sum \text{count}(t_i)}{t} \right], \text{count}(x) = x
\]  

(3)

The distribution of frequent episodes takes into account paragraph factors, so the distribution is more uniform. Considering that the dimensions represented by the frequent set may be different, if the text is divided according to all generated frequent sets, the cut subtext is excessively fragmented, and the burden of text encryption and restoration will be exacerbated. So, choose a average index select a frequent set of averaged dimensions to split the text.

**Index Construction**

The index is mainly composed of two parts, the index file and the inverted file [13]. The index file is composed of the ciphertext of the keyword set of the subfile after the split text [14]. The inverted index records the address information of the text in the cloud. Before doing the construction of the index information of this article, let’s do the relevant text feature definition work.
Construct a spatial vector of vocabulary based on the related definitions of the above vocabulary \( w_i = \{ b_1, b_2, \ldots, b_v \} \), \( p \) is the number of lexical attributes. Design specific vocabulary attribute vector \( w_i = \{ w_{fi}, if_i \} \). Where \( w_{fi} \) is used to indicate the frequency of words that \( w_i \) in the text, \( if_i \) indicates that \( w_i \) appears in the frequency of paragraphs. The following describes these two attributes one by one.

The calculation of the local mean value of the word frequency attribute of the word \( w_i \) in the paragraph is defined as formula (4):

\[
wf_i = \frac{f_i}{\max\{f_i, f_2, \ldots, f_m\}}
\]  

(4)

\( f_i \) represents the number of times the word \( w_i \), which appears in the paragraph \( p_j \). The calculation of the word frequency attribute of the text obtained by combining equation (4) is defined as shown in formula (5).

\[
wf_i = \frac{\sum_{j=1}^{m} w_{fi}}{m}
\]  

(5)

The goal of formula (5) is to average the index avoiding noise pollution as much as possible. For example, the frequency of occurrence of a word in a certain paragraph is particularly high, but in other paragraphs it is not. Therefore, the word may be noise data, which can be weakened by averaging. The second attribute indicator is \( if_i \), \( if_i \) indicates that the inverse segment of the vocabulary is calculated as (6):

\[
if_i = \log \frac{m}{tf_i}
\]  

(6)

\( tf_i \) represents the number of segments in the vocabulary \( w_i \) appearing in the text. It can be seen that the smaller the segment frequency, the weaker the ability to reflect the lexical segment. The segment frequency reflects the theme ability to a certain extent. Combining text's feature attributes with text's vocabulary's associated attributes to help implement the text's keyword set extraction algorithm is shown below.

Begin
input: file
output: keySets
using the Apriori on file get the set \( T = \{ t_i | i = 1, 2, \ldots, t \} \)
\( w = \{ \} \)
for i in T:
\( w \).append \( w \cup i \)
getting set \( e = \{ e_j | j = 1, 2, \ldots, n \} \) //getting keys set by word statistics
\( w = w \cup e \), \( w = \{ m_i | i = 1, 2, \ldots, m \} \)
for i in w:
create attribute vector \( w_i = \{ w_{fi}, if_i \} \)
if \( \alpha w_{fi} + \beta if_i \leq \varphi(file) \) //The threshold is defined by the select level \( \alpha + \beta = 0.5 \)
remove i from w
return w
End.

In the above algorithm, the keyword statistics are collected on the keyword set, and the associated attribute set is added to the keyword set. in \( w_i = \{ w_{fi}, if_i \} \), the vector space defined based on the vocabulary attribute is improved for all the keywords, and the door is defined in
\[ \alpha w_i + \beta i \leq \varphi(file) \]. Limit to help and filter redundant keyword information to complete more accurate collection of text keywords.

Next, complete the sub-text hybrid cloud storage distribution store and index build described above, using frequent set split text to get subfile set \( file = \{ f_b \mid b = 1,2,...,s \} \), \( f_g \) is a subfile, \( f \) is the size of subfile set. Now we transfer the \( file = \{ f_g \mid g = 1,2,...,f \} \) to \( file = \{ f_i \mid g = 1,2,...,f \} \). The conversion diagram is shown below as figure 2.

![Conversion diagram.](image)

Adding three flag information for subfile \( f_g \) to help restore file. The code flag is the unique hash code of file, it can prove subfiles belonging to one file, and the DF flag records the position information of the file where the subfile is located. The flag of MF indicates block status, MF=0, indicating that there are still blocks afterwards, MF=1 indicates tail block, and merge integrity constraint [15]. Since an uploaded text is to be stored separately in two cloud environments, that is, the file is divided, a pseudo-random function is used to scramble the file randomly into two text sets, \( file = \{ f_a \mid ga = 1,2,...,r \} \), \( file = \{ f_b \mid gb = 1,2,...,s \} \), \( r+s=f \). Now using filea and fileb achieve index construction algorithm.

Begin

Input: subfiles sets \( file = \{ f_a \mid ga = 1,2,...,r \} \), \( file = \{ f_b \mid gb = 1,2,...,s \} \), document D

for i in filea:
    code_i=hashcode(D)  //create a unique hashcode encoding
    DF=ga       //DF express the index of the original block information
    MF=0
    if DF=f:
        MF=1

Using the algorithm (2) get the subfile keyset, then we append code_i, DF and MF of subfile info into subfile to get subfile’, encrypt the subfile’ then send to the private cloud to get it cloud address. using address and subfile keyset create index.

Repeat it for fileb, their files send to public cloud.

End.

Retrieve and Restore

During the search, the user provides a trapdoor to the server. After the server successfully retrieves the index, it will return a series of ciphertexts that meet the conditions from the hybrid cloud. Here, the subfile set obtained by the user download is \( En = \{ e_i \mid i = 1,2,...,l \} \), \( e_i \) represents a cipher text block, \( l \) represents the size \( En \), according to the above text processing principle, first use the key to decrypt \( En \), according to each text block code standard bit grouping, divided into m group, according to DF standard bit sorting, can think of all groups as a matrix with m rows and n columns as shown in formula (7).

\[
\begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n-1} & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n-1} & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    a_{m1} & a_{m2} & \cdots & a_{mn-1} & a_{mn}
\end{bmatrix}
\]  

(7)
Each $a_{ij}$ in formula (7) represents a subfile. The line-by-line is a sub-file after the file is split. Here, the $a_{ij}$ value may is 0, indicating that the end of $a_{ij}$ is reached. m represents the number of files that meet the conditions for the query, and one line of subfiles combine in order to form a complete original document.

**Performance Test**

This article’s association attribute segmentation text search system is mainly composed of two parts, client and server, the client is mainly responsible for the construction of the index and the generation of retrieval trapdoors, the server is responsible for index maintenance and management, keyword retrieval and other work.

The experiment is divided into two phases. The first part is to analyze the accuracy of the text feature extraction method in this paper. The second part is the performance test using this article to build an index. The experimental development machine is configured as a CPU Intel Core i7, clocked at 2.8GHz, and the memory is 12 GB 1600MHZ DDR3. The operating environment version is macOS 10.12.6. The experimental data set adopts the Fudan University Chinese Corpus, which covers 500 subjects in each of five topics: computer, pedagogy, economics, art, and history. The Chinese word segmentation uses the NLPIR Chinese word segmentation system of the Chinese Academy of Sciences to construct a vocabulary with a total of 5320 words.

**Validity of Text Feature Extraction**

The first part is to test the accuracy of text feature extraction. Using this article method to obtain keyword feature vectors from experimental texts, combined with clustering k-means algorithm to perform unclassified training and learning text classification prediction. In the experiment, the five types of texts are divided into Two heaps, 2/3 training set, 1/3 validation set, respectively, the method of extracting features of this method and TF-IDF [16], information gain algorithm, CHI algorithm are related to the comparison, and the experimental data is shown in the following table 1, 2:

| Dimension drop rate (%) | this article | information gain | CHI | TF-IDF |
|-------------------------|--------------|------------------|-----|--------|
| Dimension drop rate     | 88.63        | 75.74            | 80.77 | 86.73  |

| Algorithm type | economic | art | computer | education | history |
|----------------|----------|-----|----------|-----------|---------|
| this article   | 83.36    | 84.98 | 79.62    | 85.15     | 81.36   |
| TF-IDF         | 81.16    | 80.21 | 74.42    | 79.93     | 75.44   |

Table 1 shows that the TF-IDF method and the method proposed in this paper have more advantages than the information gain and the CHI method in the dimensionality reduction method after the three dimensional clustering. Table 1 shows that information gain and CHI are not suitable for cluster analysis. Table 2 compares TF-IDF with the accuracy of text prediction in this method. Based on the prediction results, it is found that the text clustering prediction in this paper is more accurate. It also proves that the text feature extraction based on the text association attribute can extract the data of the response text feature information.

**Index Performance Analysis**

The second part of the experiment is to verify the time complexity of the index, Lucene index, full-text encryption index in the construction of the index time and index retrieval efficiency overhead. AES algorithm is selected as encryption algorithm in this experiment. In the experiment, The impact of splitting text blocks on constructing index is needed to take into consideration. The experiment selects a uniform frequent set to split the text. Then generated subfiles are respectively indexed with Lucene, the full-text encrypted index and the method are used to build the index. The index construction time and retrieval time diagram are shown:
The horizontal axis of Figure 3 shows the number of test texts, and the vertical axis shows the time to build the index. From the figure we can see that Lucene has the shortest construction time and is located at the bottom of the three lines, and the shortest time is spent, mainly because the Lucene algorithm is when constructing an index, the key word set is simply obtained according to the plain text, and there is no cryptographic component. However, both the method and the full-text encryption indexing method of this article require index cryptographic operations to construct an index. Analyzing in time, we find that the method of this article is less time-consuming than the full-text method of encrypting and building indexes. In general, the method is more advantageous in terms of ensuring the efficiency under the premise of security.

Figure 3. The time of index building.

The horizontal axis of Figure 4 is the number of test texts, and the vertical axis is the response time. The three lines in the figure respectively represent the response time of the three index structure tests. It is found that Lucene has the fastest response time because Lucene operates based on the plaintext. The operation of plaintext is much less complicated than the operation of ciphertext. Look at the method of this article. Because of the introduction of related vocabularies and vocabulary vector models, and comparison with the same full-text ciphertext search method with encryption, when the search order is less than or equal to 1000, The search time of the full-text ciphertext is less. With the increasing number of texts, the efficiency of the method used in this paper is more and more obvious than that of full-text ciphertext search. Overall, it can be compared that the method of this paper is completely superior to the search index based on full-text encryption. This is due to the importance of the relevance attributes and word vectors to the selection of index keywords.

Figure 4. The time of index research.
Conclusions

The research of file search under cloud storage has practical value. Based on this, this paper proposes a search scheme based on the association attribute to store the text stored in the hybrid cloud environment.

This paper improves the association attribute algorithm to help obtain frequent sets of text, while using frequent sets as segmentation strategies to split the text. In addition, the combination of text vocabulary paragraph vectors to select the keyword set of the text helps construct a more effective keyword set of the text, and completes the search for the hybrid storage subtext in the hybrid cloud environment according to certain strategies. Finally, through the relevant experiments to prove the feasibility and efficiency of the program.

References

[1] Vaquero, LM; Rodero-Merino, L; Caceres, J; Lindner, M. A Break in the Clouds: Towards a Cloud Definition[J]. ACM Sigcomm Computer Communication Review, 2009, (1):50-55.

[2] Xu Xiaolong, Zhou Jingyu, Yang Geng. Data Privacy Protection Mechanism for Cloud Storage Based on Data Partition and Classification[J]. Computer Science, 2013, 40(2): 98-102.

[3] Sudan, Zhou Ming-quan, Wang Xue-song, Ren Yu-zhi. Method based on least document frequency for text feature extraction [J]. CEA, 2012, (10): 164-166.

[4] John T. Kent. Information Gain and a General Measure of Correlation [J]. Biometrika, 1983, 70(1): 163-173

[5] Xu Ming, Gao Xiang, Xu Zhigang, Liu Lei. Microblog feature extraction method based on improved chi-square statistics[J]. CEA, 2014, (19): 113-117.

[6] Kamara, Seny. Encrypted Search[J]. XRDS: Crossroads, The ACM Magazine for Students, 2015, 21(3): 30-34.

[7] Wu Qi, Wan Chang-xuan. Multi-user Conjunctive Keyword Search Scheme over Ciphertext[J]. Computer Science, 2013, (9): 147-151.

[8] Chen He-feng, Lin Bo-gang, Yang Yang, Wu Yang. Chinese Keyword Fuzzy Search over Encrypted Cloud Data[J]. Information Security, 2014, (7): 69-74.

[9] Shu Xiaowei, Yang Geng, NA Haiyang. Research on parallel crypt inverted index[J]. Computer Engineering and Applications, 2016, 52(20): 14-19.

[10] Cheng-Fa Tsai, Yi-Chau Lin, Chi-Pin Chen. A new fast algorithms for mining association rules in large databases. IEEE International Conference on Systems, Man and Cybernetics, 2002, (7): 6 pp. vol. 7.

[11] He,Bo. The Algorithm of Mining Frequent Item sets Based on MapReduce[A]. Proceedings of International Conference on Soft Computing Techniques and Engineering Application: ICSCTEA 2013, September 25-27, 2013, Kunming, China. pp. 529–534.

[12] Yuan, Xiuli. An improved Apriori algorithm for mining association rules [J]. AIP Conference Proceedings, 2017, 1820(1):6.

[13] Ma Jian Zhang, Taihong, Chen Yanhong. New inverted index storage for Chinese search engine[J]. Journal Computer Applications, 2013, 33(7):2031-2036.

[14] Wu Yang, Lin Bogang, Yang Yang, Chen Hefeng, et al. Fuzzy keyword search over encrypted cloud data[J]. Computer Engineering and Application, 2015, 51(24): 90-96.
[15] Zhang Lihong, Chen Jing, Du Ruiying, et al. Data Integrity Verification Method with Secure Deduplication in Cloud Storage[J]. Computer Engineering, 2017, 43(1):32-36.

[16] Aizawa, Akiko. An information-theoretic perspective of tf–idf measures[J]. Information Processing and Management, 2003, 39(1): 45-65.