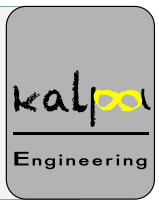
Kalpa Publications in Engineering

Volume 2, 2018, Pages 81–88

Proceedings on International Conference on Emerging Trends in Expert Applications & Security (2018)



Plagiarism Checker and Classification of Files on Cloud Using Smart Cloud

Manish Kumar Sharma[1], Shilpi Sharma[2]

Amity School of Engineering and Technology,

Amity University, Uttar Pradesh

Noida – India

[1] manish.sharma.262@gmail.com,[2]ssharma22@amity.edu

Abstract

This research paper proposes an approach to solve the issue of duplicacy in clouds. there are lots of unused and duplicate data on the internet that only consumes the limited amount of data storage these service providers can provide. If the cloud is not efficient enough it can lead to much higher cost than expected for the service providers. Another issue is that the data is stored in a server that might be present several miles away and not close to the user. This further adds to the cost and efficiency. These issues can be solved with the help of a Smart Cloud. We will take the proportions of two vectors and create a ratio which rates the document in the classification.

1 Introduction

Cloud computing is a revolutionary mechanism that has changed the approach to incorporate hardware and software outline. Cloud computing provides several benefits to its users such as simple access to their data, avoid large capital expenditures on hardware and upgrades, improved security and compliance, and so forth. Everyone from small businesses to large companies are moving towards the digital era through cloud computing. Even the majority of the smart devices these days utilize the cloud to store their user's private information. These devices are working a direct result of cloud computing [1]. There are also several applications that are running on the cloud servers. Hence, it is conceivable to state that there is a lot of identical and comparable data on the cloud which makes it difficult for users to find some information on the cloud. This issue can be explained by the grouping of information which can make the files less demanding to discover on the cloud. This solution then leads to another problem, that is, efficiency of the cloud. Recently, there has been a sudden growth of the amount of data stored onto the cloud. The total amount of digital data stored in 2013 was about 3.5 zettabytes and is expected to grow to 40 zettabytes by the year 2020. The capacity gap between the demand of data storage and production is a huge problem. It is harder to manufacture huge amounts of capacity to meet with the amount of data stored on it [2]. It is also not possible to create new data centers as it would

need a huge investments. The solution as proposed in the research paper is very simple, cost-friendly and could save huge amounts of data if implemented properly by the cloud service providers.

The main reason behind this paper is that there are lots of unused and duplicate data on the internet that only consumes the limited amount of data storage these service providers can provide. If the cloud is not efficient enough it can lead to much higher cost than expected for the service providers. Another issue is that the data is stored in a server that might be present several miles away and not close to the user. This further adds to the cost and efficiency. These issues can be solved with the help of a Smart Cloud. It is capable of handling all of these complications which with help of certain algorithms like TF-IDF algorithm, brisk-key and pronto-key algorithms [3]. A Smart Cloud uses a plagiarism checker to check the plagiarism of a document before it is uploaded onto the cloud. The server then processes the content of the document and compares with the documents already present in the cloud. If there is more than a certain percentage of duplicate data on the document, which is determined by the providers, the cloud generates a warning to the user. If the document is original and does not contain huge amounts of duplicate data, the document is uploaded on the cloud. This process does not allow similar documents to be uploaded onto the cloud over and over again, keeping the cloud efficient.

2 Literature Review

There are several algorithms and methods used to improve the functioning and efficiency of the cloud. This can be done by several ways, such as applying 3D security in the framework consisting of two phases. In the first phase, the entire data is encrypted and cannot be read without a proper key. It is then stored in the cloud. This encryption is done on the basis of availability, integrity and the confidentiality of the user. Afterwards, a critical rating is given which is calculated on the premise of these variables and protection is given on the basis of these ratings. In the second phase, the user who wants to access the data has to go through an authentication process [4].

A cloud file management system which can be used for document clustering is used to improve the management of the files present in the cloud. The system would comprise a hybrid of top-k frequent item sets and k-means. The top-k frequent item sets which is part of SHDC algorithms are used in par TF-IDF algorithm.

3 Methodology

3.1. TF-IDF Algorithm

TF-IDF stands for Term frequency-Inverse Document Frequency. It is a widely used weighting descriptive mechanism for the documents [5].

Term frequency (TF) is the amount of times a term appears in a document. It is calculated as:

$$TF = \frac{\substack{number\ of\ occurence\ of\ the\ keyword\ in \\ that\ particular\ document}}{\substack{total\ number\ of\ keywords \\ in\ the\ document}} \qquad (1)$$

Inverse Document Frequency (IDF) measures the uncommonness of a term in the whole archive. It measures the rarity of a term.

$$idf = log \frac{N}{df} \quad (2)$$

where N is the aggregate number of reports in a collection and df is the document frequency.

The concept of term frequency and inverse document frequency are consolidated, to deliver a composite weight for each term in each corpus.

$$tf\text{-}idf = tf * idf. \quad (3)$$

4 Result

This report proposes an approach to resolve the duplicate content issue that clogs up the storage in the cloud. As we already know, Cloud services are used by everyone from individual users to large organizations. A huge problem faced by the cloud is that there are a lot of duplicate files present in the cloud. An identical file can be upload on the cloud with a different name and these redundant documents occupy a lot of space in the cloud [6].

For example, that there are 5 files in a cloud as shown in the figure, most of which are identical to other files present in the cloud. Assume that the content of files given below are almost identical. The total size of all these files would add up to 1773 kb. If duplicate documents could have been avoided, the size taken up in the cloud would have been 200 kbs instead of 1773kbs. This duplicacy wastes valuable resources and storage available. This also increases the price of the maintenance and storage on the cloud.

S.NO	NAME	SIZE (KB)
1.	Name.pdf	200
2.	DuplicateName.pdf	243
3.	Codes.doc	341
4.	CodeDesign.doc	456
5.	DupliName2.pdf	533

Table. 1 Example of list of files on the cloud.

Table. 2 represents the number of files downloaded by clients in the duration of the file it had been uploaded on to the cloud. Download shows the number of uploaded files downloaded by clients whereas Days shows how many days the file had been on the cloud. The documents in the list are unclassified and is tough for the client to search for desired document which is also huge problem.

S. NO	File	Download	Days
1.	Name.pdf	55	10
2.	DuplicateName.pdf	65	8
3.	DupliName2.pdf	53	7
4.	Codes.doc	23	16

5.	CodeDesign.doc	160	17
6.	DocumentClustering.doc	45	7
7.	Security.pdf	78	14
8.	Plagiarism.doc	45	8
9.	TDIDF.doc	78	7
10..	Cosine.pdf	34	17

Table. 2 Files containing the searched word by the user.

Table.3 shows the approximate distance of the servers from the client's location. It how efficient cloud services might be while the client accesses the files. When a client tries to access a file that is uploaded on the cloud server, the cloud randomly allocates a server to the client. For example, if a client from Delhi is allocated a server that might be located in Canada that is far away from the client, it decreases the efficiency of the network and download speed decreases.

S.No	Server	Distance from Connaught Place, Delhi (km)
1.	Melbourne, Australia	10,133.6
2.	London, United Kingdom	6374.11
3.	New York, USA	19,605,2
4.	Singapore	5897.22
5.	Toronto, Canada	20361.85
6.	Moscow, Russia	5221.91

Table. 3 Distance of the servers from the client.

5.1 Proposed Approach:

5.1.1 Content Duplicacy Checker

The checker's algorithm is an algorithm which is used to compare the duplicity of the content of a document uploaded by a user with the rest of the documents in the cloud. It deals with the process of summarization of the text in a document with the help of steps below [7]:

5.1.1.1 Tokenization

It is the process of removing the white and void spaces from the corpus. The whitespaces characters are. For example, spaces, punctuations, line break. At first, the documents are complete pack of expressions with groups of stopping keywords. It is used for eliminating the white spaces and punctuation [8].

5.1.1.2. Stemming

It is the procedure of reducing the derived words to their stem word. It helps in reducing the overhead from the plagiarization checker. In general simple stemmer looks up the inflected form in a lookup table. This approach is very agile and easily handles special cases.

Sometimes it follows Suffix Stripping algorithm because it doesn't depend on lookup table.

If the word ends in 'ed', remove the 'ed'. If the word ends in 'ing', remove the 'ing'. If the word ends in 'ly', remove the 'ly'.

5.1.1.3. Removal of stop word

There are different stop words in the document which is not possible to compare, so we have to remove the stop words to it a legitimate content compare document. During document clustering, these words are eliminated to avoid special cases and thereby helping in efficient group of words which can be selected as the stop words for a specific purpose. The most common stop words are: **the**, **is**, **at**, **which**, **on** and so on. After removing these, the remaining data is saved to the database for comparing purpose.

5.1.2 TF-IDF Algorithm:

1) Word **w** in the processing document **d** where **d € D.**

2) Wd = fw,d * log(|d|/fw,D)

3) Repeat above steps for all the words in **d** to calculate **Wd** for every word."

4) Words with high **Wd** imply that they are important words in **d."**

Where, 'D' is the set of documents on the cloud, 'd' is the document which is processing under algorithm, 'W' is the word in 'd' which is processing in the algorithm, 'Wd' is the weight of the word in 'd', 'Fw,d' is the number of word 'w' in document 'd' and 'Fw,D' is the number of word 'w' in set of documents D.

5.1.3 Enhancing the search and accessing documents from cloud

To enhance the search and accessing of documents from cloud, we use the Brisk-Key algorithm, which is used to perform grouping of the data available in the cloud.

To solve this problem we use a Brisk-Key algorithm which is capable to perform classification of the data present in the cloud. The criteria used for the grouping of the documents are the date on which the files were uploaded and the number of times it has been viewed by the users. These variables are then used to assess a 'ratio' which is utilized to rate the file [9].

5.1.4 Steps for the proposed model using Brisk-Key Algorithm [11]:

Step 1: The first step is to declare variables, 'ratio', 'period' and 'downloads'. The variable 'period' shows the number of days the file has been on the cloud since it was uploaded on the cloud by a user. The variable 'downloads' gives the value of number of times the file has been accessed.

Step 2: To access the N^{th} document, the variable ratio (period, downloads) is used to evaluate the value as follows:

ratio(N)=downloads(N)/ period(N) (5)

The above function would run until the end and would calculate the ratio of the N^{th} document and then later save it to a database.

Step 3: The documents are then classified into an organized order using bubble sort algorithm. This puts the higher ratio documents on the top of the list and returns the list to the client.

S. NO	File	Downloads	Period	Ratio
1.	TDIDF.doc	78	7	11.142
2.	CodeDesign.doc	160	17	9.411
3.	DuplicateName.pdf	65	8	8.125
4.	DupliName2.pdf	53	7	7.571
5.	DocumentClustering.doc	45	7	6.428
6.	Plagiarism.doc	45	8	5.625
7.	Security.pdf	78	14	5.571
8.	Name.pdf	55	10	5.500
9.	Cosine.pdf	34	17	2.000
10.	Codes.doc	23	16	1.437

Table. 4 Files classified by their ratios

Step 4: The clients would have to select a document based on the value of the variable ratio calculated in the previous step. After the document is accessed by the client, the algorithm shows the number of servers on which the files are to be accessed from.

Step 5: The proposed will then request the client to allow the cloud server to find out the location of the user. After which, the cloud detects the clients' coordinates and the suitable server for the client to use.

Step 6: The servers are then set in order according the distance of the server from the user's location using bubble-sort algorithm

S. No	Server	Distance from Connaught Place, Delhi (km)
1.	Moscow, Russia	4352.98
2.	Singapore	5897.22
3.	London, United Kingdom	6374.11
4.	Melbourne, Australia	10,133.6
5.	New York, USA	19,605.20
6.	Toronto, Canada	20361.85

Table. 5 Distance of the servers from Connaught Place, New Delhi

Step 7: If the server does not have full access to the document, it will return the value of 0. There would be no changes in the variable downloads(n) and would remain the same which means that the file's number of downloads would remain the same. If the server does have full access to the document, it will return the value of 1 and add it to the variable downloads(n). The change in the variable download is shown as below:

$$downloads(n)=downloads(n)+1 \qquad (6)$$

If the value returned is 1, then it means that the file has been downloaded and it is reflected on to the server. The value is then updated in the database.

5 Conclusion

In this paper, we proposed an approach through which we can reduce the amount of duplicate documents that are uploaded on the cloud using TF-IDF algorithm. If this approach is implemented properly, it will allow the system to be more robust and would make the cloud plagiarism-free. With the inclusion of more peculiar and better algorithmic approach, it would enable faster and efficient result. It will also block the user's document if the new document had more than the set percentage of plagiarized content as that of pre-existing document over the cloud. For ranking the documents in each cluster, we applied the cosine similarity between every pair of documents in each cluster. Using this, we calculate the similarity factor of each document and finally rank their values. This ranking of documents will help the user to get the necessary documents at the beginning of each cluster and reduced his search process. This work can further be extended by considering those documents which are not parts of the initial clusters formed by the proposed approach because of strong association rule, to make either new **clusters** or part of the existing clusters which may be of user interest [10].

References

[1] Jayalekshmi, M. B., and S. H. Krishnaveni. "A study of data storage security issues in cloud computing." Indian Journal of Science and Technology 8.24 (2015).

[2]http://www.techradar.com/news/internet/data-centre/world-could-run-out-of-storage-capacity-within-2-years-warns-seagate-vp-1278040/2

[3]https://en.wikipedia.org/wiki/Plagiarism_detection

[4] Ngnie Sighom, Jean Raphael, Pin Zhang, and Lin You. "Security Enhancement for Data Migration in the Cloud." Future Internet 9.3 (2017): 23.

[5] Bafna, Prafulla, Dhanya Pramod, and Anagha Vaidya. "Document clustering: TF-IDF approach." Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on. IEEE, 2016.

[6] Omar, K., Alkhatib, B., Dashash, M., & Alhassan, F. (2 16). The Implementation of Using Medical Ontologies in Plagiarism Detection. International Review on Computers and Software (IRECOS), 11(3), 232-238.

[7] Harnik, Danny, Benny Pinkas, and Alexandra Shulman-Peleg. "Side channels in cloud services: Deduplication in cloud storage." IEEE Security & Privacy 8.6 (2010): 40-47.

[8] "Box.net lets you store, share, work in the computing cloud". Silicon Valley Business Journal. December 6, 2012.

[9] Wu, Weili, Hui Xiong, and Shashi Shekhar, eds. Clustering and information retrieval. Vol. 11. Springer Science & Business Media, 2013.

[10] Li, Jin, et al. "A hybrid cloud approach for secure authorized deduplication." IEEE Transactions on Parallel and Distribute