A Bloom Filter-Based Data Deduplication for Big Data

Shrayasi Podder and S. Mukherjee

Abstract Big data is growing at an unprecedented rate with text data having a large share and redundancy is a technique to ensure availability of this data. Large growth of unstructured text data hinders the primary purpose of the big data rendering the data difficult to store and search. Data compression is a solution to optimize the use of the storage space for big data. Deduplication is the most useful compression techniques. This paper proposes a two-phase data deduplication mechanism for text data. In the syntactic phase, a combination of clustering and Bloom Filter is used. In the semantic phase, a combination of SVD and WordNet synset is employed. Experimental results show the efficacy of the proposed system.

Keywords Deduplication · Bloom Filter · Clustering · SVD · WordNet

1 Introduction

A large amount of data to the tune of terrabytes are generated every day by various mediums. Eighty percent of generated data is claimed to be unstructured and in the text format [1], making data management a difficult, time-consuming, and complex task. The big data is primarily characterized by the rate of growth of the data. This data is stored in various storage mediums including cloud. To contain such a massive amount of data, storage continues to grow at an explosive rate (52% per year) [2, 3]. By the end of 2020, the size of the total generated data will surpass 30 zettabytes (ZB) as per a very conservative estimation [4, 5].

However, generating new data is just one quantum of the real problem of the growth of the data. Unprecedented growth of the data is also contributed to other,

S. Podder
Institute of Engineering and Management, Kolkata, India
e-mail: pshrayasi@gmail.com

S. Mukherjee (✉)
DIST, CEG, Anna University, Chennai, India
e-mail: msaswati@auist.net

M. L. Kolhe et al. (eds.), *Advances in Data and Information Sciences*, Lecture Notes in Networks and Systems 38, https://doi.org/10.1007/978-981-10-8360-0_15

more complex, data handling mechanisms employed. To handle data easily, data needs to be at the location of the user, and in the digital world today, a user may be in any geographic location. While with the growth of the Internet, it is possible to serve the data to a user anywhere, bandwidth and other issues pose as serious challenges. To resolve this problem, typically data would be duplicated and distributed to various geographic locations for ease of access, thereby relieving the need of bandwidth usage and networking across large geographical regions. However, this causes a unique problem of redundant duplicated data occupying more and more storage space. Redundancy in the big data is contributing proactively to the rapid increase in the size of this data. Although new forms of storage such as Cloud employ different techniques to improve storage efficiency, it is a costly method, especially in the scenario where the demand of a specific data reduces after an initial period. To alleviate this problem, such duplicates must be managed, which entails ensuring that the unnecessary duplicates are detected and deleted; a challenging task, given the nature and size of the big data.

Typically, the technique of deduplication is widely used to identify and remove similar contents from various data storages, thereby reducing cost and saving space in data centers [6]. Deduplication is a very effective method that claims to save up to 90–95% of storage. Even though data deduplication technique had evolved as a simple storage optimization technique for secondary storages, it is widely adapted in larger storage areas like cloud storage. Different data deduplication is widely used by various cloud storage providers like Dropbox [7], Google Drive [8]. Deduplication is becoming a popular method for not only in backups and archives, but also in primary data centers. The major benefit of deduplication is saving disk space, and it also reduces the search space for searching such data.

While applying deduplication mechanisms is difficult in numeric data, it is particularly challenging in case of text data since, besides having identical data blocks, text data may also be similar in the semantic sense. Besides identifying and removing identical data, an efficient deduplication method for text data must identify and eliminate semantically similar data as well.

To handle deduplication of text data effectively, this research proposes a two-stage mechanism that can efficiently deduplicate a large amount of text data. The first phase is the syntactic similarity phase, where two documents are subjected to a process of detecting similarity by comparing the set of words in the documents. A combination of clustering and a proposed Bloom Filter called Updatable Bloom Filter is used in this phase. This mechanism can identify identical documents. Semantic similarity is handled in the second phase. This is the process of detecting semantic similarity between two documents by not only comparing the set of words in the documents but also comparing their synonyms. A combination of SVD and WordNet synset is used in this phase.

The paper is organized as follows. Section 2 investigates the existing work in the area of deduplication, clustering, and Bloom Filter. Section 3 discusses the proposed method. In Sect. 4, algorithms are described. Section 4 shows the efficacy of the system using the experimental results with Sect. 5 bringing the conclusion.

2 Background Details and Related Work

Over the years, many efficient solutions to the deduplication problem by measuring document similarity and by using Bloom Filter were proposed. Su et al. [9] suggested the method of using an integer Bloom Filter in cloud storage. The algorithm used for hashing in Bloom Filter is Secured Hash Algorithm (SHA-1). A number of hash functions have been defined. Data is divided into chunks that are processed in different clusters and are checked for duplicates using Bloom Filter in each cluster. As a result, unique data are written on the storage. Merlo et al. [10] suggested a method of deep learning, called Self-Organizing Maps (SOM). da Cruz Nassif et al. [11] used clustering algorithms like k-means and k-medoids. K-medoid is the process of taking median of data points in a cluster. K-means is the process of taking mean of data points in a cluster. After this step, hierarchical clustering techniques like centroid-based, average-link based were used to identify similar documents. Combinations of k-means and k-medoid with hierarchical clustering techniques were performed and the one with higher accuracy was chosen. Jiang et al. [12] suggested a simple method of using a feature to check if the documents are similar. Feature can be a word or a set of words. Feature can either appear in both documents, in any one document or, in none of the documents. Based on the number of features matched, similarity score is assigned between two documents. Pires et al. [13] proposed a vector space model which involved machine learning techniques namely k-nearest neighbors, random forests, and support vector machines. These techniques use cosine similarity as the distance measure. Semantic analysis has also been performed which included word co-occurrence, matching noun phrases, WordNet synonyms, etc. Gemmel et al. [14] implemented an idea of removing duplicate entries from a spreadsheet of names and addresses. The authors used affine gap distance, which is a variation of Hamming distance to check whether two records are similar. Historically, most deduplication-related publications focus on a narrow range of topics: maximizing deduplication ratios and read/write performance. Further, existing research work on document deduplication is either syntactic or semantic according to the need of the application. Machine learning techniques used are largely time-consuming and many used mechanisms cannot process very large documents. Attempts to modify clustering techniques to improve time have been performed. The proposed model aims at creating a new model for deduplication using machine learning and semantic techniques, attempting to optimize the limitations found by measuring the similarity of the data both syntactically and semantically.

3 Proposed Approach

In the proposed method, we employ a two-phase deduplication. The first phase performs deduplication based on the syntactic similarity, and the second phase is based on semantic similarity.

3.1 Syntactic Phase

To perform deduplication using syntactical similarity, it is required to find all pairs of documents that are identical to each other and replace one with a pointer to the other. This process can be recursively applied to all the documents for complete deduplication of a set of documents. However, in a big data environment, there will be a large number of documents in a data center and the method of applying such mechanisms consumes time as well as CPU cycles. This research proposes to optimize using two approaches. First, the documents are clustered such that each cluster contains lesser number of documents. Since clustering groups similar documents together, similarity need not be measured between documents across clusters. To create the initial clusters, we apply a k-means clustering. *K-means* clustering is dependent on the initial seed and its efficacy depends on the robustness of initial seed selection. To improve this method, we employ fractionation algorithm for obtaining the initial k seeds. The corpus is broken into n/m buckets each of size $m > k$. An agglomerative algorithm is applied to each of these buckets to reduce them by a factor of ν. Thus, at the end of the phase, we have a total of $\nu \cdot n$ agglomerated points. The process is repeated by treating each of these agglomerated points as an individual record. This is achieved by selecting one of the documents within an agglomerated cluster. The approach terminates when a total of k seeds remain. These seeds are used as centroids in standard k-means clustering algorithm in order to determine good clusters.

A document is added to the cluster having the smallest cosine similarity value between the document and the corresponding centroid of that cluster. For every iteration, cosine similarity is calculated between the documents and centroids to decide whether a document should be moved to another cluster or not. The document which lies closest to the mean value of all the documents in a cluster is made the new centroid of that cluster. The above processes are repeated until the centroids do not change.

During the formation of the clusters, deduplication is performed by using proposed Forgetful Updatable Bloom Filter (FUBF). A Bloom Filter (BF) is a space-efficient probabilistic data structure that can be used to test whether an element is a member of a set.

A basic BF uses an array of m bits. The whole array is initially set to 0. When a document is to be inserted, it is hashed using a fixed number of hash functions, k. Therefore, the BF can be represented using parameters (m, k). Each hash output of a BF maps to a bit in the filter, which is then set to 1. To check for the presence of a document, the same set of hash functions are used to verify if all the mapped bits are already 1. In a Bloom Filter, it is possible to get false positives (i.e., a document not present may appear to be present) but no false negatives are possible (i.e., it always returns the right answer for a document not present) [15]. However, research shows that there have been ways to keep the rate of false positives to an acceptable level (almost close to 0) by manipulating parameters [16, 17]. Hence, Bloom Filters can be employed to check for the presence of a duplicate in a cluster during the process of

creating the cluster, thereby avoiding altogether the incidence of adding an identical duplicate to a cluster.

While BF is an efficient way, it gives rise to two following problems that lead to scalability issue [18].

1. Nothing is forgotten by the BF. Hence, the bits set by older operations cannot be deleted.
2. Parameters (m, k) need to be fixed at the time of creation of the Bloom Filter.

As a result, once the Bloom Filter fills up, it is hard to "scale up" or "move" its elements to another BF with different parameters. Hence, a BF provides more and more false positives as it fills up since more bits are set to 1 by the incoming documents.

We propose to use a Forgetful Updatable Bloom Filter (FUBF) that employs the techniques of Forgetful Bloom Filter [18]. The proposed FUBF uses three filters for the representation of the documents in each cluster:

1. A current documents Bloom Filter;
2. A old documents Bloom Filter;
3. A new documents Bloom Filter.

All these Bloom Filters are equal in size and identical in their use of hash functions. Before a new document is to be inserted in a cluster, it is checked for the presence of its duplicate in the cluster using FUBF. All the three filters of the FUBF of the cluster are checked for this purpose. If the tested document is present in at least one of the three constituent filters, a duplicate is found and the new document is not inserted. However, if it is not found in any of the three filters, it is inserted only into the new and current Bloom Filters, but not in the old Bloom Filter. This way, the FUBF purges the filters of the older documents giving way to add more and more documents in a cluster, ensuring scalability.

At the end of the complete syntactic similarity phase, we obtain k clusters of documents having no syntactically similar documents. We perform semantic similarity on these clusters.

3.2 Semantic Phase

In the semantic phase, semantic similarities between documents are considered. We propose to use a variant of Latent Semantic Analysis (LSA) to check the semantic similarity. Documents in each cluster from syntactic phase are further divided into concept-based clusters using LSA that employs Singular Value Decomposition (SVD) method. We propose to apply LSA to obtain those documents that contain the same concept and calculate similarity between all pairs of such documents. If the similarity measure for a pair of documents is greater than a pre-defined threshold, these documents are subjected to the next level of semantic checking where WordNet synset is used for obtaining sentence level concept-based similarity. In this level, each

document from a pair is split into sentences. Nouns and named entities are used to compare a sentence with sentences belonging to the other document in the pair. If the targeted words of two sentences have the same tag, then the sentences are considered to be similar. If all the sentences are similar, then the documents are considered as duplicates.

4 Experimental Setup and Results

The proposed method is tested against four different datasets as follows: a set of 6,000 documents which has details about different countries with 2,000 duplicates; a set of 10,000 documents having jokes that has 3,000 duplicates randomly injected and a set of 25,000 documents having jokes that has 10,000 exact and semantic duplicates; a set of 10,000 books from Gutenberg Web site with 2,000 exact duplicates. The proposed method is tested against two existing approaches. The first is constraint-based k-means clustering algorithm which is

Table 1 Deduplication using the proposed model

Dataset name	No. of data	No. of duplicates present	No. of duplicates found	F-measure
Country dataset	6,000	2,000	1,809	0.94
Joke dataset	10,000	3,000	2,521	0.91
Joke dataset	25,000	10,000	9,107	0.95
Gutenberg books	10,000	2,000	1,823	0.94

Table 2 Deduplication using constrained k-means

Dataset name	No. of data	No. of duplicates present	No. of duplicates found	F-measure
Country dataset	6,000	2,000	1,409	0.70
Joke dataset	10,000	3,000	1,921	0.66
Joke dataset	25,000	10,000	7,897	0.78
Gutenberg books	10,000	2,000	1,389	0.76

Table 3 Deduplication using chunking

Dataset name	No. of data	No. of duplicates present	No. of duplicates found	F-measure
Country dataset	6,000	2,000	1,609	0.87
Joke dataset	10,000	3,000	2,121	0.76
Joke dataset	25,000	10,000	8,970	0.88
Gutenberg books	10,000	2,000	1,598	0.84

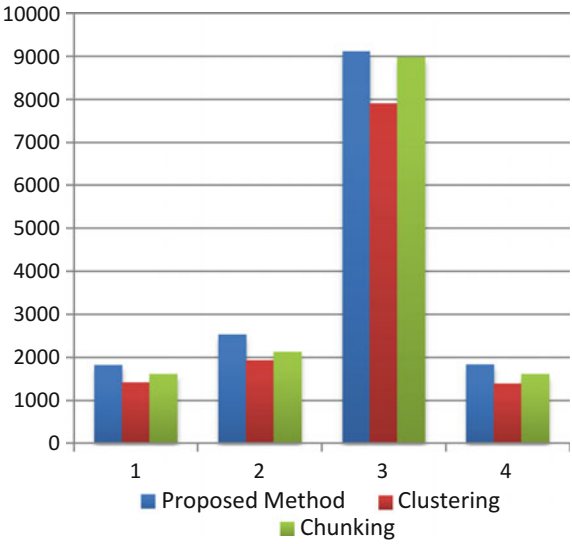

Fig. 1 Improved results of the proposed system

a deduplication method proposed by Hu et al. [19]. The second is deduplication using chunking method as suggested in [20]. These algorithms are applied on the same datasets and the corresponding results are compared. Table 1 shows the result of applying our proposed two-phase method on all four datasets. Table 2 shows the results of constrained k-means clustering method on all the datasets using 200 constraints. Table 3 shows the results of applying chunking for deduplication over the same datasets. F-measures are calculated for each method. Figure 1 shows the performance comparison of the three methods. It could be noted that the F-measure of the proposed model is higher than the other methods. It is evident that the extra levels of Bloom Filter during clustering and LSA and WordNet-based methods after clustering have increased the recall of the process, thereby improving F-measure.

5 Conclusions

From the results, it is clear that the performance of the proposed two-phase deduplication method is better than the pure clustering or chunking-based approach. The proposed method can be further improved by implementing concept-based clustering for the semantic phase.

References

1. CWADN, http://www.computerweekly.com/
2. Eaton C, Deroos D, Deutsch T, Lapis G, Zikopoulos P (2012) Understanding big data. McGraw-Hill Companies
3. https://www.smartfile.com/blog/the-future-forecast-for-cloud-storage-in-2018/
4. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html
5. Reed DA, Gannon DB, Larus JR (2012) Imagining the future: thoughts on computing. Computer 45
6. Deduplication, http://en.wikipedia.org/wiki/Data_deduplication
7. https://www.dropbox.com/
8. https://www.google.com/drive/
9. Su YH, Chuan HM, Wang SC, Yan KQ, Chen BW (2014) Quality of service enhancement by using an integer bloom filter based data deduplication mechanism in the cloud storage environment. In: IFIP international conference on network and parallel computing. Springer, Berlin, pp 587–590
10. Su YH, Merlo P, Henderson J, Schneider G, Wehrli E (2013) Learning document similarity using natural language processing. Linguistik Online 17(5)
11. da Cruz Nassif LF, Hruschka ER (2013) Document clustering for forensic analysis: an approach for improving computer inspection. IEEE Trans Inf Forensics Secur 8:46–54
12. Jiang J-Y, Lin Y-S, Lee S-J (2014) A similarity measure for text classification and clustering. IEEE Trans Knowl Data Eng 26:1575–1590
13. Pires CE, Nascimento DC, Mestre (2016) Applying machine learning techniques for scaling out data quality algorithms in cloud computing environments. Appl Intell 45:530
14. Gemmell J, Rubinstein BIP, Chandra AK. Improving entity resolution with global constraints. https://arxiv.org/abs/1108.6016
15. Bose P, Guo H, Kranakis E, Maheshwari A, Morin P, Morrison J, Smid M, Tang Y (2008) On the false-positive rate of bloom filters. Inf Process Lett 108(4):210–213
16. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426
17. Wikipedia (2015) Bloom filter. https://en.wikipedia.org/wiki/Bloom_filter
18. Subramanyam R (2016) Idempotent distributed counters using a forgetful bloom filter. Clust Comput 19(2):879–892
19. Hu G, Zhou S, Guan J, Hu X (2008) Towards effective document clustering: a constrained *K*-means based approach. Inf Process Manag 44:1397–1409
20. Tolic A, Brodnik A (2015) Deduplication in unstructured-data storage systems. Elektroteh Vestn 82(5):233