Societal Transformation: AI and Big Data
Journal Vol: 3(1): 19-26,2025
DOI: 10.20547/aibd.253102

A Bloom Filter-Based Approach for Textual Data Deduplication in Data Warehousing

Muhammad Ali[1]

Abstract: Data duplication is one of the core issues in data warehouses pertaining to the quality of data. By finding and removing duplicate data, you can decrease the amount of space needed to store your data. For smooth, efficient, and fast analysis the duplicated data needs to be filtered out before using it for further process. However, very few research has been done on data deduplication for textual data. Realizing this gap, this paper presents analysis of bloom filters to detect duplicates in textual data. The paper discusses the challenges and the need for textual data deduplication. Existing literature on the topic is classified. Then, bloom filter based textual deduplication pipeline is presented. The proposed approach is evaluated with different parameters to investigate the impact on false positives and storage efficiency. It has been found that an optimal number of hash functions can reduce duplication along with maintaining a low memory overhead. These findings also highlight tradeoff between accuracy, scalability and computational cost. Hence, the proposed approach presents a promising solution for large scale textual data deduplication in data warehouse.

Keywords: *bloom filter, duplicate detection, data warehouse, textual data*

1. Introduction

In recent years, big data has gained prominence. A large volume of data is generated daily through internet, IoT devices, social media and transaction management systems. Yet, the volume of data generated is increasing day by day [8]. To manage such data, the emergence of frameworks such as Hadoop, Apache Spark, pig, map/reduce, hive have been witnessed [16].

Because of the massive volume of data, it is essential to identify duplicates during data updates. The data deduplication process is very important for deriving actionable insights, decision making and providing a competitive advantage [6]. *Data warehouse* is a large repository that accumulates data from different sources.

This work proposes a bloom filter-based approach for duplicate detection in textual data. Bloom filters and their various types are widely used as fast and space-efficient probabilistic data structures [5]. The Bloom filter can be defined as employing hash functions to identify membership in a set[1]. Bloom filters can be used for large number of applications. It is useful in the situation when one wants to perform set membership tests quickly especially for big data archives. Some of the most widely used applications of bloom filter are spell-checker,

[1] *NED University of Engineering and Technology, Karachi*

differential file updating, distributed network caches, and textual analysis. Bloom filter is actually a probabilistic method with a predefined error rate [2].

The contributions of the paper are as follows. We first identify the need for textual data deduplication in data warehouse. The major challenges pertaining to data deduplication is discussed. Relevant literature / taxonomy pertaining to data deduplication is also presented. The paper identifies bloom filter as a novel solution for handling duplicates. A data processing pipeline is presented, therefore. The rest of the paper is organized as follows. The next section discusses the major challenges for data deduplication. Then, literature review is presented. This is followed with an introduction to bloom filter and its application in data deduplication. Then, the proposed methodology is discussed followed by results. The paper concludes with discussion on future work.

2. Challenges and the need for textual data deduplication

Data deduplication is a big challenge in data warehouse because data arrives from multiple sources and it is possible that two fields that are same have different naming conventions, format because of coming from different sources [7]. This challenge makes analysis difficult and may hamper decision making. On the contrary, two records with some non-essential making may be tagged as duplicates despite they are different. To avoid such problems, data deduplication is required.

To address the data deduplication challenge, this paper proposes a bloom filter-based solution for detecting duplicates. The paper primarily focuses on textual data. Following points signify the importance of data deduplication in textual data:

- If the text data is duplicate, it can lead to incorrect reporting, misleading insights. A single source of truth is necessary for reliability [18].
- Duplicate data increases the cost of storage [17]. In addition, there are unnecessary delays in query processing and optimization
- Data deduplication can improve pipeline efficiency [19]
- Inaccurate data can lead to problems in marketing, finance and fraud detection

However, there are significant challenges in textual data deduplication. There is textual variability such as capitalization, spacing, typos, abbreviations and synonyms. There can be contextual sensitivity issues. Finally, data deduplication algorithms must be scalable. Let's discuss some of these challenges:

- *Data variability and inconsistency:* Data can be represented in different formats [20]. For instance, Karachi can be written as KHI. There can be typing mistakes and different spellings for the same word. There can be the use of synonyms. The data can be in different languages and can have phonetic similarities
- *Ambiguity in matching:* There can be partial duplicates. Also, false positives and false negatives are possible. The context of the data must also be considered
- *Scalability and performance* [21]*:* The computational complexity of the data deduplication algorithm should not be a high order polynomial
- *Unstructured data:* As the data is unstructured, there may not be any primary key or identifiers concept

- *Matching algorithms:* The selection of the matching algorithms and threshold requires careful consideration. In addition, the domain of the data (such as medical, legal, education) may require different matching algorithms specialized for those domain
- *Selection of the victim record:* Which record to keep and which to be deleted requires significant expertise or human intervention, and may be dependent on organizational rule

Mitigating these challenges require advanced NLP techniques and AI. This paper employed bloom filter for handling data deduplication.

3. Related Work

There has been significant interest in data deduplication techniques for data warehouse system. Interested readers are referred to [12] for detailed discussion. There are different approaches to data deduplication proposed in literature. Amongst them is the similarity-based approach that is based on identifying similarity between data items [9]. [6] has presented data deduplication techniques and incremental upload in Hadoop data warehouse system. [8] has proposed an architecture for data deduplication for data lakes. A data deduplication workflow is designed. The general process remains same. The data is ingested, deduplicated and stored. A six step data deduplication framework has been proposed in [9]. In [10], a secure data deduplication scheme for textual data in cloud environments have been proposed. Another data deduplication scheme has been proposed based on data compression and encoding in [11]. Fuzzy logic has been used in [13] for data deduplication in data warehouse. Machine learning has been used for data deduplication in [14].

Figure 1 provides the taxonomy of various approaches for data deduplication in data warehouse system. Amongst the various data deduplication techniques include rule-based approaches, probabilistic/ fuzzy matching approaches, machine learning approaches, deep learning approaches and graph-based approaches. The rule-based approach relies on pre-defined rules/ heuristics to identify duplicates in the data. Probabilistic approaches consider similarity scores instead of strict equality. The machine learning approach is based on identifying patterns in the data for duplicates. Deep learning approaches learn representations of records for deduplication tasks. Finally, graph based approaches are based on connected component analysis, community detection, and graph embeddings to consolidate duplicates.

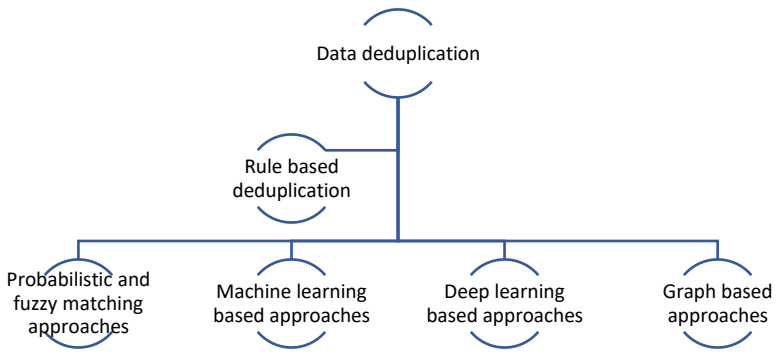

Figure 1: Taxonomy of various approaches for data deduplication

We also quote some work related to Bloom filter. The authors in [2] proposed an approach based on Bloom filters to search for related passages in a monograph. The approach is based on Bloom filters of all the possible words in each text of a monograph. Then, as the next step, it computes the normalized dot product of all pairs of them. The outcome of the dot product is actually a similarity measure. It must be noted that the higher the value of the dot product, it will be more likely that the text will have similar content.

In our earlier work, we introduced bloom filter for identifying duplicates for generic data in data warehouse [15]. This work has been extended in this research to propose a novel pipeline and bloom filter approach for identifying textual duplicates.

4. Preliminaries

A *bloom filter* is a data structure for the probabilistic representation of a set X that supports membership queries i.e., a query that checks whether element A is part of set X. It is a space-efficient data structure with a minor rate of false positives.

Bloom filter has the advantage of space and query time efficiency. It performs more efficiently compared to common algorithms. It uses a hash table to find whether an element is part of a set or not. The false negatives are impossible in Bloom Filter. However, the Bloom filter has one major shortcoming is that it can produce false positives.

Several Bloom filter variants have been proposed that address some of the limitations of the original structure, including counting, deletion, multisets, and space efficient bloom filters.

5. Methodology

This section discusses the methodology for data deduplication. We will discuss the processing pipeline as well as the bloom filter-based approach used to identify duplicates.

5.1. Processing pipeline

The processing pipeline comprises various steps as shown in Figure 2.

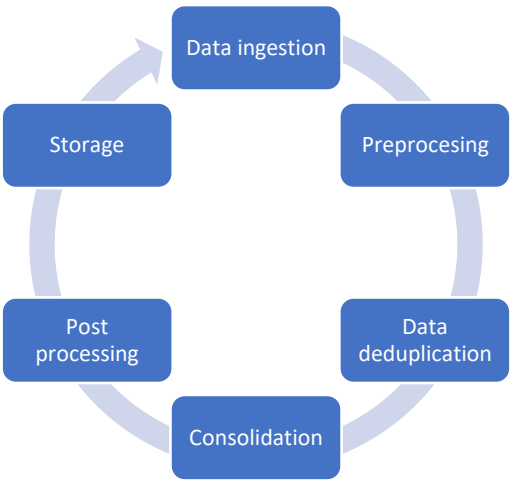

Figure 2: Processing pipeline for data deduplication

In the first step, data is acquired from different sources. This includes databases, logs, APIs, CSVs etc. In the *preprocessing* phase, various techniques are applied. This includes case-normalization, white space removal, punctuation removal, stop-word removal, stemming, lemmatization and standardization. Then, *data deduplication* is performed. For this purpose, bloom filter is applied to remove the duplicate. After duplicates are identified, they are consolidated. They can be performed automatically or based on human review. *Post-processing* steps can be performed to improve accuracy of data deduplication. Then, the data is stored in the data warehouse.

Even though the pipeline involves extensive steps for preprocessing, these basic steps are essential to ensure that textual variations don't cause any issue in duplicate detection phase. The fact that textual data is noisy with variations may lead to false positive matches. Such steps are standard in natural language processing and are performed automatically. However after the duplicates are detected, record consolidation may be performed using automated merging techniques or human review (as a future work). Finally, these steps are lightweight and modular enabling the system to adopt to different domains scalable to larger datasets.

5.2. Using bloom filter for identifying duplicates

Listing 1 shows the bloom filter based approach for duplicate detection. In the first step we Initialize the array with the *m* number of bits. Then, we pass the words through hash function. Then, take the mod with *m* of hash output. If the value is 0 on the array index, then increment the array index location of array.

The choice of a simple hash function was used to establish a baseline. However, the theory of bloom filter is agnostic to the nature of hash function. The only condition is independence and uniform distribution. The robustness of the bloom filter is already established in literature. However, incorporating more sophisticated and domain-specific hash functions may optimize the performance further such as for the specific domain of heterogeneous data warehouse environment. It is also noted that optimal duplicate detection requires careful selection of parameters. This is in particular with respect to the number of hash functions used in the bloom filter.

Listing 1: Bloom filter approach for textual data deduplication

```
1: K = bit(m)
2: hash = h(K)
3: i = hash mod m
4: If the value is 0 on the array index
{
        i+=1
```

```
}
```

6. Results & Analysis

Figure 3 shows the results on various hash functions Vs. the false positive rate. We first pass the input with a single hash function. It was producing false positives initially. At this stage, the system is producing a relatively high number of false positives. This is due to the limited discriminatory power of single hash function. Then, we increased the hash functions. In this case, the false positives rate also decreased till at a particular point. After that when we increase the number hash function the false positive results get increased again. This means the trend didn't continue indefinitely. Beyond a certain threshold, further increase in the number of hash functions lead to a rise in false positive again.

We applied the solution to a case study of spell checker. For the spell checker, we pass the word to the initialized bloom filter and check whether the word exists in it or not. With this, we can use it with any language, as we just need to initialize the bloom filter with the vocabulary of the language.

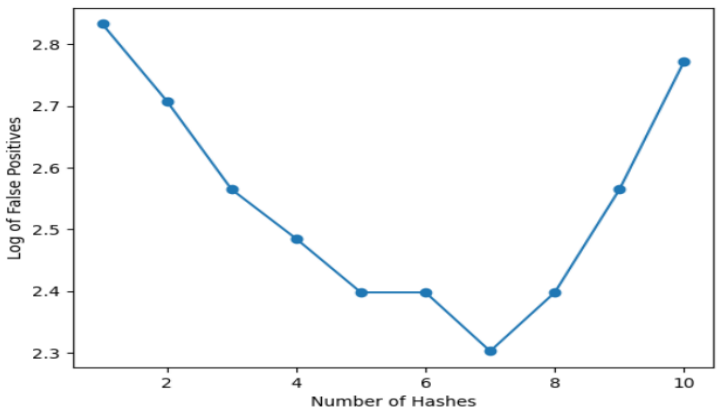

Figure 3: Number hashes vs false positives analysis

It can be concluded that the number of hash functions gives the optimized result till at a particular point after that it again produces the false positive. As the results indicate, a smaller number of hash functions increases the likelihood of false positives while a higher number of hash functions may cause collisions leading to rise in false positives again. These results reflect that an upper limit on the number of hash functions should be chosen wisely to maintain optimal data deduplication performance. Finally, these parameters should be selected based

on the dataset peculiarities and application requirements. The parameter selection is not a technical standpoint, but an important factor for large scale deployment of data deduplication solutions.

7. Conclusion

This paper discusses an approach for textual data deduplication in data warehouse. With our analysis, we can conclude that the bloom filter can be improved with the increase of hash functions. However, the upper limit for the number of hash functions should be set very effectively. Because this will optimize the performance of the bloom filter. Despite the experiment conducted on a smaller dataset, the dataset doesn't limit the general applicability of bloom filter. The concept has been well-studied in the domain of data structure, have scalability properties, and the same applies in this discourse irrespective of the dataset size. The employment of a compact dataset enabled controlled experimentation and analysis of how false positives vary with multiple hash function parameters. The larger dataset is anticipated to show similar trends with greater scale in memory usage and throughput. Finally, the current literature confirms that bloom filter grows logarithmically with dataset size in terms of memory consumption.

There are certain limitations associated with the proposed approach. The current approach is based on exact text matching after normalization. Therefore, the proposed approach doesn't address semantic equivalence. In domain such as medical/ legal studies, synonyms, abbreviations or partial matching will remain undetected by lexical approaches such as the one proposed in this study. This is a limitation of this study and future work can be done on addressing semantic similarity by use of techniques such as word embeddings, ontologies or contextual language models. There are certain other avenues where this research can be pursued further. The analysis of the proposed approach needs to be performed on a larger scale. We have used a simple hash function. More sophisticated hash functions can be used.

8. References

[1] Bloom, B.H., 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, *13*(7), pp.422-426.

[2] Blustein, James, and Amal El-Maazawi. "Bloom filters. a tutorial, analysis, and survey." *Halifax, NS: Dalhousie University* (2002): 1-31.

[3] John Crickett https://www.linkedin.com/pulse/coding-challenge-53-bloom-filter-spell-checker-john-crickett-bqerc?trk=public_post

[4] Jiang, M., Zhao, C., Mo, Z. and Wen, J., 2018. An improved algorithm based on Bloom filter and its application in bar code recognition and processing. *EURASIP Journal on Image and Video Processing*, *2018*, pp.1-12.

[5] Tarkoma, S., Rothenberg, C.E. and Lagerspetz, E., 2011. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials*, *14*(1), pp.131-155.

[6] Julakanti, S.R., Sattiraju, N.S.K. and Julakanti, R., 2022. Incremental Load and Dedup Techniques in Hadoop Data Warehouses. *NeuroQuantology*, *20*(5), pp.5626-5636.

[7] Wangikar, V.C., Deshmukh, S.N. and Bhirud, S.G., A Semi-Automated Record De-Duplication Technique for a Data Warehouse Environment.

[8] Hlavačka, J., Bobák, M. and Hluchý, L., 2024. Big Data Deduplication in Data Lake. *Acta Polytechnica Hungarica*, *21*(11).

[9] Azeroual, O., Jha, M., Nikiforova, A., Sha, K., Alsmirat, M. and Jha, S., 2022. A record linkage-based data deduplication framework with datacleaner extension. *Multimodal Technologies and Interaction*, *6*(4), p.27.

[10] Ghassabi, K. and Pahlevani, P., 2024. DEDUCT: A Secure Deduplication of Textual Data in Cloud Environments. *IEEE Access*.

[11] Miri, A. and Rashid, F., 2019, October. Secure textual data deduplication scheme based on data encoding and compression. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 0207-0211). IEEE.

[12] Costa, G., Cuzzocrea, A., Manco, G. and Ortale, R., 2011. Data de-duplication: A review. *Learning structure and schemas from documents*, pp.385-412.

[13] Ananthakrishna, R., Chaudhuri, S. and Ganti, V., 2002, January. Eliminating fuzzy duplicates in data warehouses. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases* (pp. 586-597). Morgan Kaufmann.

[14] Revanth, N., Varun, R., Harshitha, E., Rubel Angelina, J.J. and Subhashini, S.J., 2024. Data Deduplication using Machine Learning. *Grenze International Journal of Engineering & Technology (GIJET)*, *10*.

[15] Rizwan, S., Adil, S.H. and Islam, N., 2019, November. Detecting Duplicates in Real-Time Data Warehouse Using Bloom Filter-Based Approach. In *International Conference on Intelligent Technologies and Applications* (pp. 762-771). Singapore: Springer Singapore.

[16] Syed Noorullah Shah. 2023. A Comparative Study of Big Data Frameworks. Societal Transformation: AI and Big Data Journal

[17] Fan, W. and Geerts, F., 2012. Data Deduplication. In *Foundations of Data Quality Management* (pp. 69-91). Cham: Springer International Publishing.

[18] Maddodi, S., Attigeri, G.V. and Karunakar, A.K., 2010, November. Data deduplication techniques and analysis. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology* (pp. 664-668). IEEE.

[19] Afreen Khalfe. 2024. "Data Deduplication Strategies: Reducing Storage and Improving Query Performance", https://talent500.com/blog/data-deduplication-strategies-reducing-storage-and-improving-query-performance/

[20] "Understanding Textual Data Challenges in the Tech Industry". 2024. *The Tech Artist* https://thetechartist.com/textual-data-challenges/?utm_source=chatgpt.com

[21] "Scalable Data De-duplication Techniques for Storage Optimization". 2023. *MomentsLog*, https://www.momentslog.com/development/architecture/scalable-data-de-duplication-techniques-for-storage-optimization