Deduplication in Databases using Locality Sensitive Hashing and Bloom filter

[1]Sadhik M S,[2]Eldo P Elias,[3]Surekha Mariam Varghese
[1,2,3] Dept. Computer Science and Engineering, Mar Athanasius College of Engineering
Kothamangalam, India

Abstract : **Duplicates in databases represent today an important data quality challenge which leads to bad decisions. Deduplication is a capacity optimization technology that is being used to dramatically improve storage efficiency. In large databases, sometimes find ourselves with tens of thousands of duplicates, which necessitates an automatic deduplication. It can reduce the amount of storage cost by eliminating duplicate data copies. In proposed system, introduce an effective duplicate detection method for automatic deduplication of text files and repeated strings. In this paper, we propose a similarity-based data deduplication scheme by integrating the technologies of bloom filter and Locality Sensitive hashing (LSH), which can significantly reduce the computation overhead by only performing deduplication operations for similar texts. In the proposed system check the strings or text in the repository that are similar. If they are similar, then remove the string and maintain only one copy of the data. Locality Sensitive Hashing and bloom filter methods provide better results than those of known methods, with a lesser complexity.**

IndexTerms **- Deduplication, Bloom filter, Levenshtein Distance.**

I. INTRODUCTION

The limitless heterogeneity of internet resources and the recent advances in the areas of computation and communications have forced the overloading of digital libraries and data warehouses. Extracting useful information from these datasets is very time consuming. Recently the focus of text analysis has been shifting toward short texts such as microblogs, search queries, search results, ads, and news feeds. Similarity measurement between short texts is a fundamental task and can be used for various applications including text clustering and text classification [2]. To reduce resource consumption, many cloud storage services employ a deduplication technique. Here mainly focus on data deduplication. It is often called as intelligent compression or single-instance storage. It is a process that eliminates redundant copies of data and reduces storage overhead. Data deduplication methods ensure that only one unique instance of data is retained on storage media.

As database systems are more and more common place, data cleaning is going to be the cornerstone for correcting errors in systems which are accumulating vast amounts of errors on a daily basis. Deduplication is the process of identifying duplicate files during the discovery process and removing them from further processing and analysis. Most of the duplicate detection systems are available today offer various algorithmic approaches for speeding up the duplicate detection process. A duplicate file is an exact copy of another file. Deduplication is necessary for many situations involving electronic documents because multiple identical documents are a typical feature of large record sets. Deduplication is an important method that is very effective in reducing the storage in databases. The proposed system provides a personal assistance for doctors as well as for patients. The system is only used for reference not a replacement for the doctors.

Though various deduplication techniques have been proposed and used, no single best solution has been developed to handle all types of redundancies. Considering performance and overhead, each deduplication technique has been developed with different designs considering the characteristics of data sets, system capacity and deduplication time. In the proposed system, mainly focus on hospital databases. In hospitals the patient details that are stored in the form of abstracts. It contains the details of the patients and their diseases. Many patients have a similar disease and here there is a possibility of deduplication.

For the duplicate detection step, use the methods such as Locality Sensitive Hashing (LSH) and bloom filter concept. By some measures, LSH has been around almost 20 years. LSH is found in a diverse set of academic research areas including digital signal processing, information retrieval, statistics, machine learning, and data mining. As such, searches of the literature reveal a fluid definition of the term LSH and related items. Distance measure becomes important in this world of alternate representations of documents. There are Euclidean distance measures, cosine distance, and so on. On a basic two dimensional plot, this can be as simple as measuring the distance between two points. It can get more complex when working with vectors in higher dimensions. Alternatively, when working with strings or even just bits, other distance measures can be leveraged. Examples there include Levenshtein distance or Hamming distance. So in the proposed system, LSH uses Levenshtein distance algorithm for string comparison. In LSH, use *Levenshtein distance algorithm* for string comparison. The *Levenshtein distance* is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. Bloom filter is another method that is used in the proposed system. Bloom Filter, a probabilistic data structure that uses multiple hash functions to store data in a large bit array was introduced in 1970 by Burton H. Bloom. It is mainly used for membership queries when dealing with large data sets.

II. RELATED WORK

The automatic elimination of duplicate data in a storage system, commonly known as deduplication, is increasingly accepted as an effective technique to reduce storage costs. Thus, it has been applied to different storage types, including archives and backups, primary storage, within solid-state drives, and even to random access memory.

Jinfeng Liu in 2017, propose a novel secure similarity-based data deduplication scheme by integrating the technologies of bloom filter and content-defined chunking, which can significantly reduce the computation overhead by only performing deduplication operations for similar files. A Bloom filter is a data structure which manages memory-efficiently, whether an element is present in a set. The space advantages are more difficult to sum up; again it depends on the error rate you're willing to tolerate. It also depends on the potential range of the elements to be inserted; if it is very limited, a deterministic bit vector can do better. If you can't even ballpark estimate the number of elements to be inserted, you may be better off with a hash table or a scalable Bloom filter.

Locality-sensitive hashing (LSH) reduces the dimensionality of high-dimensional data. LSH hashes input items so that similar items map to the same "buckets" with high probability (the number of buckets being much smaller than the universe of possible input items).LSH differs from conventional and cryptographic hash functions because it aims to maximize the probability of a "collision" for similar items. Hashing-based approximate nearest neighbor search algorithms generally use one of two main categories of hashing methods: either data-independent methods, such as locality-sensitive hashing (LSH); or data-dependent methods, such as Locality-preserving hashing (LPH). Locality-sensitive hashing has much in common with data clustering and nearest neighbor search. In [20], Indyk and Motwani proposed a hashing based high-dimensional approximate similarity search scheme called Locality-Sensitive Hashing (LSH). This scheme is linearly depends on the size of the dataset. Instead of partitioning method, it uses several hashing methods to hash the points to increase the probability of collision for objects which are similar than for those which are not. Then near neighbors can be determined by hashing the queried point and retrieving those elements stored in buckets that contain those points. This LSH scheme is applied when the points are given in binary Hamming space. The fundamental drawback of LSH is that it is fast and simple only when the points are available in the Hamming space. It is further discussed that the algorithm can be used so that data can be extended to be in hamming distance, but it is achieved at time complexity cost and in increased error (by a large factor).

Broder et al. [11] designed an LSH scheme based on Jaccard similarity of sets which uses Minhash functions to take different permutations to calculate Jaccard similarity for sets [9]. Several variations have been proposed to improve the performance of this LSH scheme in [12]. In [23], Panigrahy et al. proposed an LSH scheme which is entropy-based. It minimizes the number of hash tables required, by looking up a number of query offsets in addition to the query itself. In addition to considering the bucket corresponding to the query, buckets corresponding to perturbed versions of the query also considered. Unfortunately, while the space requirements are reduced, the query time is considerably increased.

In 2003, Peter Christen [3] proposes Standard Blocking (SB) method clusters records into blocks where they share the identical blocking key [14]. A blocking key is defined to be composed from the record attributes in each data set. An example of a blocking key is the first four characters of a surname attributes. A blocking key can also be composed of more than one attribute, for example, a postcode attribute could be combined with an age category attribute. There is a cost-benefit trade-off to be considered in choosing the blocking keys [15]. If the resulting blocks contain a large number of records, then more record pairs than necessary will be generated, leading to an inefficiently large number of comparisons. For example, using a gender attribute as blocking key puts all the available records into two very large blocks. On the other hand, if the blocks of records are too small, then true record pairs may be missed, therefore reducing linkage accuracy (sensitivity).

Canopy Clustering with TFIDF (Term Frequency/Inverse Document Frequency) forms blocks of records based on those records placed in the same canopy cluster. A canopy cluster is formed by choosing a record at random from a candidate set of records (initially, all records) and then putting in its cluster all the records within a certain loose threshold distance of it. The record chosen at random and any records within a certain tight threshold distance of it are then removed from the candidate set of records. The number of record pair comparisons resulting from canopy clustering is O(fn2/c) [16] where n is the number of records in each of the two data sets, c is the number of canopies and f is the average number of canopies a record belongs to. The threshold parameter should be set so that f is small and c is large, in order to reduce the amount of computation. However, if f is too small, then the method will not be able to detect typographical errors. In 2005, Andrew McCallum [6] provides empirical evidence that using canopies for clustering can increase computational efficiency by an order of magnitude without losing any clustering accuracy. The basic clustering approach use here is Greedy Agglomerative Clustering. In order to perform clustering in this domain, we must provide a distance metric for the space of bibliographic citations. A powerful choice for measuring the distance between strings is string edit distance, as calculated by dynamic programming using different costs associated with various transformation rules.

III. PROPOSED SYSTEM

Technique for the automated deduplication of data and text files in the database is proposed. The proposed system is based on the methods like Locality Sensitive Hashing (LSH) and using the concept of bloom filter. The data mainly related to the hospital database.

In this proposed system, the abstract data of the patient is newly entered into the database. The preprocessing module can take place. In this module, the given data can be tokenized and then the tokens that are lemmatized. The similarity checking module checks the similarity of each data in the repository by using different methods such as LSH and Bloom filter.

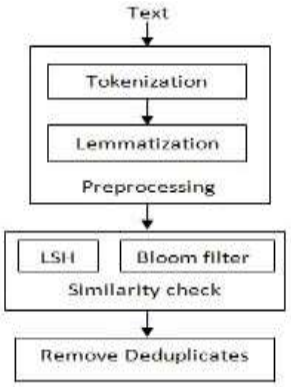

Fig 1: Proposed system architecture

The proposed system has 3 stages – Preprocessing, similarity checking and Deduplication process. The proposed architecture is shown in Figure 1. It shows how each of the phases is related to its predecessor phases. The first step is preprocessing the given text. This is to provide a representative sample of the data to deduplicate, or of the type of these data. In preprocessing step, analysis of the sample is performed. A tokenizer divides the text into a sequence of tokens, which roughly correspond to "words". Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. Lemmatization is a more methodical way of converting all the grammatical/inflected forms of the root of the word. Lemmatization uses context and part of speech to determine the inflected form of the word and applies different normalization rules for each part of speech to get the root word.

In the second module, similarity checking is the process of identifying the similarity between the disease descriptions in the database. In LSH, distance measure become important in this world of alternate representations of documents. There are Euclidean distance measures, cosine distance, and so on. On a basic two dimensional plot, this can be as simple as measuring the distance between two points. It can get more complex when working with vectors in higher dimensions. Alternatively, when working with strings or even just bits, other distance measures can be leveraged. Examples there include Levenshtein distance or Hamming distance. In the proposed system, Levenshtein distance algorithm is used to check the similarity between strings.

Bloom filter is another method that is used in the proposed system. Bloom Filter, a probabilistic data structure that uses multiple hash functions to store data in a large bit array. It is mainly used for membership queries when dealing with large datasets. It reduces O (n) search time to constant time. The search accuracy depends on the size of Bloom Filter (m) and a number of hash functions(k) and the major advantage of multiple hash functions is that search accuracy is improved. This approach is very effective when speed matters more than space.

In the third module, deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. As soon as the data is uploaded, there is a high chance for duplicate chunks occurrence in the repository. So it is mandatory for identifying and removing the duplicates in the repository. The identification of duplicate is done using Levenshtein Distance Algorithm (LDA). With the help of bloom filter system reduces the overall search time.

IV. CONCLUSION

Though various deduplication techniques have been proposed and used, no single best solution has been developed to handle all types of redundancies. The objective of this work is to improve the storage utilization in databases. From the experiments using the methods like LSH and bloom filter improve the efficiency of the system. It can provide efficient deduplication in databases. In LSH, use Levenshtein distance is used to identify the same data in the repository. The experimental results show that system is able to achieve reduction in storage size and overall search time. From the observation, it could be concluded that the presence of bloom filter and semantic analysis greatly improves the system performance.

REFERENCES

[1] Ibrahim M N, Amolo-Makama Ophéli. Fast Semantic Duplicate Detection Techniques in Databases,Journal of Software Engineering and Applications, 2017, 10, 529-545

[2] Jinfeng Liu a, Jianfeng Wanga,b, Xiaoling Taoc and Jian Shend. (2017) Secure similarity-based cloud data deduplication in Ubiquitous city, Pervasive and mobile computing.

[3] Bhagyashri, Kelkar, A. and Manwade, K.B. (2012) Identifying Nearly Duplicate Records in Relational Database, International Journal of Computer Science and Information Technology & Security , 2, 514-517.

[4] Baxter, R., Christen, P. and Epidemiology, C.F. A Comparison of Fast Blocking Methods for Record Linkage. Proceedings of Workshop Data Cleaning, Record Linkage, and Object Consolidation, Washington DC, August 24-27 2003, 25-27.

[5] Aizawa and Oyama, K. A Fast Linkage Detection Scheme for Multi-Source Information Integration. Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration, Tokyo, 8-9 April 2005, 30-39.

[6] Yan, S., Lee, D.W., Kan, M.-Y. and Lee, C.G. Adaptive Sorted Neighborhood Methods for Efficient Record Linkage. Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries , Vancouver, 18-23 June 2007, 185-194.

[7] McCallum, Nigam, K. and Ungar, L.H. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, Boston, 20-23 August 2000, 169-178.

[8] Ramos, J. (2003) Using Tf-idf to Determine Word Relevance in Document Queries.

[9] Metha, Kadhum, Alnoory, Musbah and Aqel, M. (2011) Performance Evaluation of Similarity Functions for Duplicate Record Detection. Master's Thesis, Middle East University, Beirut.

[10] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," Proc. International Conference on Distributed Computing Systems (ICDCS), pp. 617–624, 2002.

[11] P. Indyk and R. Motwani, "Approximate nearest neighbor: towards removing the curse of dimensionality", Proc. Symposium on Theory of Computing, 1998.

[12] A. Broder et al., "Min-wise independent permutations", Proc. Theory of computing, ACM Symposium, New York, USA, pp. 327-336, 1998.

[13] M. Hernandez and S. Stolfo. Real-world data is dirty: data cleansing and the merge/purge problem. Journal of Data Mining and Knowledge Discovery, 1(2), 1998.

[14] M. A. Jaro. Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. Journal of the American Statistical Society, 84(406):414–420, 1989.

[15] Bianco, G.D., Galante, R. and Heuser, C.A. A Fast Approach for Parallel Deduplication on Multicore Processors. 26th Symposium on Applied Computing SAC'11, TaiChung, 21-25 March 2011, 1027-1032.

[16] Raghavan, H. and Allan, J. Using Soundex Codes for Indexing Names in ASR Documents. Proceeding SpeechIR '04 Proceedings of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval at HLT-NAACL 2004, Boston, 6 May 2004, 22-27.

[17] Christen, P. A Comparison of Personal Name Matching: Techniques and Practical Issues. 6th IEEE International Conference on Data Mining-Workshops , Hong Kong, 18-22 December 2006, 290-294.