A Technical Solution for Data Indexing Technology in Plagiarism Checker

Yatheendra KV[1], Dr. Sudhakara Arabagatte[2]

[1]Research Scholar, College of Computer Science, Srinivas University, Mangalore, India
Email: *yatheendra72[at]gmail.com*

[2]Professor, College of Computer Science, Srinivas University, Mangalore, India
Email: *sudhakara.mysore[at]gmail.com*

Abstract: ***Plagiarism is considered a serious academic and ethical offense, as it undermines the values of originality, honesty, and integrity in academic and creative work Plagiarism detection is the process of locating instances of plagiarism within a work or document. The widespread use of computers and the advent of the Internet have made it easier to plagiarize the work of others. Most cases of plagiarism are found in academia, where documents are typically essays or reports. Here I am working with a team to support their process. Counterfeiting identification is the way toward finding examples of written falsification inside a work or archive. The far - reaching utilization of PCs and the appearance of the Internet have made it less demanding to appropriate the work of others. Most instances of written falsification are found in the scholarly world, where records are regularly articles or reports. Here working with a group to bolster their procedure. We are facing many challenges to develop this sort of software. So, mainly the data indexing methods are very interesting in this case. Here we are exposing how data indexing methodology works using 'Taylor series' formula in cloud - based storage for data indexing.***

Keywords: Indexing, Encryption, Data Sequence, Search Key

1. Introduction

Data indexing technology is the process of creating an index or catalog of the content of a collected of texts or other, to facilitate efficient searching, retrieval, and analysis. This is the process of creating an index or database of searchable terms or keywords that can be used to quickly find specific pieces of text. There are several ways to do text indexing, but one common method is using search engines or specialized software tools. In the market they have indexing methodologies like Apache Lucene, Elasticsearch, and Solr etc. ., but they have some character limitations. Ex: Languages are Urdu, Arabic or Persian are Right to Left languages, you can't make sure others are providing best accuracy in this, so avoiding such sort of issue we can make it our own indexing protocol. Eventually, our indexing process involves identifying the key concepts, terms, or entities that are important for describing the content of the texts, and creating an index that URL maps these concepts to the locations in the texts where they occur. This allows users to search for specific words or phrases and retrieve relevant texts quickly, rather than having to read through the entire collection. Text data indexing is a fundamental technique in many information retrieval systems of plagiarism checking. The core characters and supporting characters, in each language containing more than 125 characters and referenced by minimum four length of Unicode characters, Example in letter 'A' Unicode is U+0041. Here, if all text data come with all characters, the indexing level in a single cloud storage platform goes to a very high level, if huge numbers of indexing level will kill the searching operation during the plagiarism checker process. And it will take more operational expenses and delay. On the other end we have to find and avoid some special non - indexing characters. These non - indexing characters will create exception status and sometimes it will create crashes in operation during implementation. Commonly non - indexing refers to ': ? > < * \ /' but in the case of all characters we can cover almost 22*150 (including special) characters should be a level of initial indexing methods. The indexing will be done by organizing a tree structure of data which finally refers to the position of a cloud - based file path. A single text file is divided into multiple selected sequences according to our algorithm, and sequence removed duplication to avoid indexing ambiguity. This sequence is passed to the index number of levels until destination. Finally, the indexed file will store the cloud data path to refer to the same file path for referencing and the indexed results must return to be accurate and relevant. Our code executes approximately 20 million sequences in 100 - 120 seconds, which means fast and indexing successive rate is 99.99%. we reached the best time and space complexity as our indexing expectation. The best indexing process we have done using the Taylor series is the polynomial formula explanation.

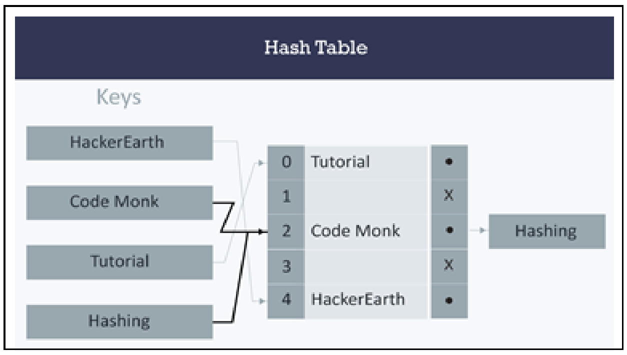
2. Experimental Method

A single file data can be split into several sequences. Ex: if you have 100 kb of text data may split into more than 3000 sequences, each sequence is a combination of consecutive characters. In a language letter 'A' is finding his Unicode character is 'U+0041', if you are running 64 bit of your cloud server then you can divide Unicode '0041' to 16 bits binary base 2 (OC85) converter level. Means 0 (decimal 00) => 0000000000000000, C (decimal 83) => 0000000001010011, 8 (decimal 72) => 0000000001001000 and 5 (decimal 69) => 0000000001000101. After combining all binary sequences, we get 64 digits (same as 64 bit of your cloud server data transfer bus) of value. Finally, u will get mod%4 operation using all 64 binary digits. We will get binary sequences 0000 0000 0001 0000, 0000 0010 0010 0011, 0000 0001 0000 0000 and 0000 0011 0000 0000. Later if you convert decimal from signed 2's complement with mod%10 operation you will get values 0C85 (input) to 6769 (output). We can conclude her search key is 0C85, indexed data is 6769, and sequence is 16 bits binary values.

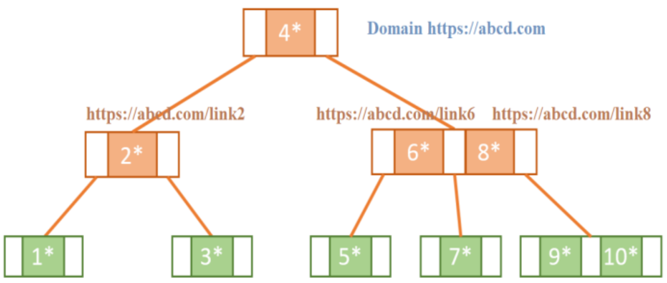
3. Other Indexing Structure

In plagiarism process data indexing, an indexing structure is a data structure sequence used to store and organize data for efficient search and retrieval. It is used in plagiarism applications such as plagiarism search engines, databases, and file systems. Here we are used four types of indexing structures plagiarism checking application explained below:

Hash table: A hash table is an indexing data structure that uses a hash function to map sequence keys to values. It provides constant - time average - case performance for search operation from large index of data, here hash index divided in to three different layers called sequency key, storage buffer and hashing, internally all three layers are mapped with proper data and values.

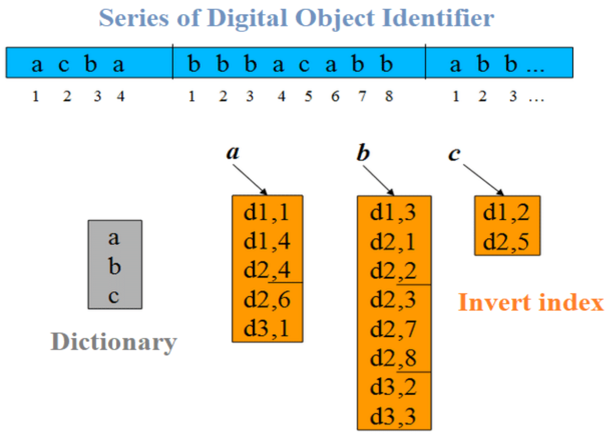


B- tree: A B - tree is a balanced tree data structure that is commonly used in databases and file systems. It allows for efficient search, insertion, and deletion operations.



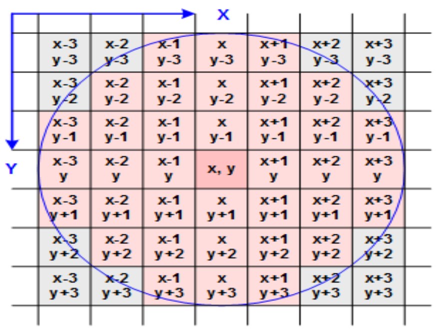
In our plagiarism indexing methodology mainly B - tree we are exposed in link indexing methods, we have here more than 300 billion of links data indexed in hashing point, but separately all links are indexed to b - tree methods for avoiding indexing ambiguity. In main domain https: //abcd. com referenced to other sub links, same other sub links referenced others until ends of link from same domain.

Inverted index: An inverted index is an indexing data structure used in search engines to store the mapping between words and the documents that contain them. It allows for efficient full - text search operations.



In invert indexing we mainly used for Digital Object Identifier (DOI) article tracking purpose, In our plagiarism system all links segregated by two variants one is absolute links other is DOI. The absolute links are already used in B - tree indexing methods and same for DOI indexing we can expose the invert indexing methodology for identification and tracking total article available in our data base for avoiding duplicate article download and index.

Pixel index: A pixel index is a data indexing structure used in databases to store the presence or absence of digital values in rows and columns like X and Y axis. It allows for efficient boolean operations such as AND, OR, and NOT. The choice of indexing structure depends on the type of data being indexed and the operations that need to be performed on it.


X
Y
x-3 y-3, x-2 y-3, x-1 y-3, x y-3, x+1 y-3, x+2 y-3, x+3 y-3
x-3 y-2, x-2 y-2, x-1 y-2, x y-2, x+1 y-2, x+2 y-2, x+3 y-2
x-3 y-1, x-2 y-1, x-1 y-1, x y-1, x+1 y-1, x+2 y-1, x+3 y-1
x-3 y, x-2 y, x-1 y, x, y, x+1 y, x+2 y, x+3 y
x-3 y+1, x-2 y+1, x-1 y+1, x y+1, x+1 y+1, x+2 y+1, x+3 y+1
x-3 y+2, x-2 y+2, x-1 y+2, x y+2, x+1 y+2, x+2 y+2, x+3 y+2
x-3 y+3, x-2 y+3, x-1 y+3, x y+3, x+1 y+3, x+2 y+3, x+3 y+3

This sort of indexing method is mainly described in the image indexing process, during image indexing all image can be extracted into number of pixels referenced by RGB color coding. Indexing can be done according to X and Y axis direction.

4. Objectives

Data cleaning: Data cleaning refers to the process of identifying and correcting or removing errors, inconsistencies, and inaccuracies from a dataset. It is an important step in the data analysis pipeline as the quality of the data directly impacts the accuracy and reliability of the insights obtained from it. The data cleaning process typically involves major steps in data indexing in plagiarism checking. This involves identifying missing values, duplicates, and outliers in the dataset. This is often done using statistical analysis or visualization tools. Once errors have been identified, they can be corrected by either imputing missing values or removing outliers. Then it converts data into a common format, such as converting dates into a standard format or normalizing text data to remove inconsistencies.

Data modeling: Data modeling is the process of creating a conceptual representation of data and its relationships to facilitate efficient data storage, retrieval, and analysis. It involves identifying the entities, attributes, and relationships that are important in the context of a specific problem or application domain. Data modeling defines the entities and their relationships without including details of how they are implemented. Further it specifies the entities, attributes, and relationships in a more concrete and implementation - independent way. In physical data model that specifies the details of how the data is stored, including data types, keys, indexes, and other implementation - specific details. Data modeling is a crucial step in the process of designing and building effective databases and data - driven applications. It helps ensure that the data is organized and structured in a way that enables efficient retrieval and analysis, and that it supports the business requirements of the application or problem domain.

Index Testing: Index testing is a testing technique that is used to evaluate the performance of a text data index. Plagiarism indexes are data structures that are used to speed up data retrieval operations in a database. Index testing is performed to ensure that indexes are working correctly and that they are providing the expected performance benefits. The primary goal of index testing is to ensure that the database queries are being executed efficiently and with high performance. The testing process in plagiarism involves executing a set of predefined queries on the database and measuring the response times. The response times are compared against predefined performance criteria, and any performance issues are identified and addressed. Index testing can be performed manually or using plagiarism customized automated testing tools. This tool can help to speed up the testing process and provide more accurate and consistent results. It is crucial to perform index testing regularly to ensure that the database is functioning optimally and delivering the expected performance benefits.

Sequence Combination

Data sequence refers to the order or arrangement of data elements or items in a dataset for plagiarism data indexing purpose purely operated by machine learning algorithm. Here mainly the order of data elements is important and can have a significant impact on how the data is analyzed or processed. For example, in a time - series dataset, the sequence of data points represents the order in which the measurements were taken over time. In natural language processing, the sequence of words in a sentence can be crucial in determining the meaning of the sentence. In data science and machine learning, sequence data is often analyzed using sequence models, which are specialized algorithms that can handle the temporal or sequential nature of the data. Some of important data sequences is segregated in Plagiarism are:

1) 5 words sequence (Normal Data).
2) 14 words sequence (Normal Data).
3) Quotes sequence
4) Reference sequence.

The sequence is an important aspect of many datasets, and understanding the order or arrangement of data elements is often crucial in the analysis and processing of the data. Sequence models are commonly used to analyze and extract insights from sequence data, and the analysis of data sequences can help to solve a wide range of problems in similarity check processing.

5. Results and Discussion

According to our indexing process, Taylor series is the polynomial, and it is a function of an infinite sum of sequence terms. Each successive sequence term will have a more exponent sequencing degree than the indexing term. A data reference is always pointing to a search key variable, then the sequence is pointing to a data reference variable similarly it's a triangle workflow of all referencing paths. When the formula enters an iterative process the Taylor series algorithm creates an index of the text data using encryption method, which includes the key terms and keywords they have automatically identified. This typically involves creating an inverted index, which maps terms to the search key where they appear. This process works like A=>B, B=>C and C=>A. In all search key, data reference and sequences workflow under Taylor series flow, here f (x) assigning all set of differentiable function [f (x) = f (a) + f' (a) (x − a) + [f″ (a) /2! (x − a) 2] +…+], where function (f called search key), neighbourhood number (a called data reference) and composite value (x called sequence). Finally, the f (x) search method holds all three operations for finding searchable value within a short time during the plagiarism checking process.

In other hand objectives involves data cleaning, data modeling and indexing refers to different level on indexing terms, in our plagiarism process data may contains text, images, video only. Video may be consisted by links indexing methods but on the other hand text and image are main part of similarity text models while similarity process. All the process may be executed by huge and complex machine learning algorithms, it may be executed under best time, lesser space and good quality levels of processing.

Time Complexity:

Time complexity is a measure of the amount of time required by an algorithm to solve a indexing problem as a function of the input sequence size. It is often expressed using Big O notation, which describes the upper bound of the time required by an algorithm as the input size approaches infinity. The time complexity of an algorithm is important during indexing because it helps to determine the efficiency of the algorithm and whether it is feasible to use for a given problem size. An algorithm with a lower time complexity is generally more efficient than an algorithm with a higher time complexity. For example, consider a simple algorithm that sorts an array of n sequence (1000 sequence) using bubble sort, while n is a level of indexing to reach final path. The time complexity of this algorithm is O (n^2), which means that the worst - case time required by the algorithm grows quadratically with the input size. This means that as the input size increases, the time required by the algorithm will increase much faster than the size of the input itself.

Space Complexity:
Space complexity is a measure of the amount of memory required by an algorithm to solve a problem as a function of the input size. It is often expressed using Big O notation, which describes the upper bound of the amount of memory required by an algorithm as the input size approaches infinity. The space complexity of an algorithm is important because it helps to determine the amount of memory required by the algorithm and whether it is feasible to use for a given problem size. An algorithm with a lower space complexity is generally more memory - efficient than an algorithm with a higher space complexity. For example, consider a simple algorithm that computes the factorial of a number ‘n’ using recursion or iteration. The space complexity of this algorithm is O (n), which means that the amount of memory required by the algorithm grows linearly with the input size. This means that as the input size increases, the amount of memory required by the algorithm will increase proportionally to the size of the input itself. Here in plagiarism 100 KB files are divided into 1000 sequences and 5 level of destination path, each sequence is a combination of maximum 100 characters, so our calculation is 100 * 1000 * 5 = 5, 00, 000 bytes required, finally in 4807 KB for big O (n).

Indexing Quality: Indexing quality refers to the accuracy and completeness of an index, which is a data structure used to optimize data retrieval operations. An index is typically created on one or more columns of a database table to allow fast access to the data based on specific criteria, such as a particular value or range of values in the indexed columns. The quality of an index is determined by several factors, including its selectivity, uniqueness, and efficiency. All these to produce 99.99% accuracy while in similarity indexing performance. An index is considered selective if it can filter out many rows from the table based on the indexed columns. This means that the index can significantly reduce the amount of data that needs to be scanned to retrieve the required data, resulting in faster query performance. An index is considered unique if it ensures that each index entry corresponds to a unique row in the table. This is important to prevent duplicate data from being returned in query results and to maintain data consistency. An index is considered efficient if it minimizes the amount of disk space required to store the index and the amount of memory required to perform index lookups. This is important to ensure that the index does not consume excessive system resources and to maintain good overall system performance. In addition to these factors, the quality of an index can also be impacted by factors such as the data distribution, the data types of the indexed columns, and the workload characteristics. To ensure high - quality indexing, it is important to carefully design and evaluate the indexes used in a database. This involves selecting appropriate indexed columns, tuning index parameters, and regularly monitoring index usage and performance. Proper indexing can significantly improve the performance of database queries and overall system efficiency.

6. Conclusion

From the above technical information finally, we conclude that our regional language indexing methodology is one of best data indexing technology in our plagiarism checking process. Large number of data sequences will be indexed with less duration compared to other open - source indexing technology. This sort of complex work done by Taylor series logic and security part used binary encryption sequences, both mentioned in our solution index encryption part. Above we discussed different modes of objectives called data cleaning, data modeling and index testing. Other way sequence combination shows different types of sequence structure, so finally the operation can be executed under time and space complexity. It gives the best Big O model to achieve 99.99% performance during the plagiarism process.

References

[1] J. Li, Z. Xu, Y. Jiang, and R. Zhang, “*The overview of big data storage and management. cognitive informatics cognitive computing* (icci*cc) „” in IEEE 13th International Conference on, (pp.510 - 513, 2014.

[2] C. Liu, R. Ranjan, X. Zhang, C. Yang, D. Georgakopoulos, and J. Chen, “*Public auditing for big data storage in cloud computing - , ” in A Survey. Computational Science and Engineering (CSE)*, 2013 IEEE 16th International Conference on, (pp.1128 - 1135)., 2013, Dec.

[3] H. Tan, W. Luo, and L. M. Ni, “*Clost: A hadoop - based storage system for big spatio - temporal data analytics*., ” in Proceedings of the 21st ACM International Conference on Information and Knowledge Management (pp.2139 - 2143). New York, NY, USA: ACM., 2012.

[4] W. Zhou, C. Yuan, R. Gu, and Y. Huang, “*Large scale nearest neighbors search based on neighborhood graph*, ” in Advanced Cloud and Big Data (CBD), 2013 International Conference on, pp.181–186, Dec 2013.

[5] H. Nakada, H. Ogawa, and T. Kudoh, “*Stream processing with bigdata: Sss - mapreduce*, ” in Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, pp.618–621, Dec 2012.

[6] T. Chardonnens, “*Big data analytics on high velocity streams*, ” Master’s thesis, University of Fribourg (Switzerland), June 2013.

[7] F. Amato, A. De Santo, F. Gargiulo, V. Moscato, F. Persia, A. Picariello, and S. Poccia, “*Semtree: An index for supporting semantic retrieval of documents*, ” in Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on, pp.62–67, April 2015.

[8] Cambazoglu BB, Kayaaslan E, Jonassen S, Aykanat C (2013*) “A term - based inverted index partitioning model for efficient distributed query processing”*. ACM Trans Web 7 (3): 1–23. doi: 10.1145/2516633. 2516637

[9] Bast H, CelikikM (2013) *“Efficient fuzzy search in large text collections”*. ACM Trans Inf Syst 31 (2): 1–59. doi: 10.1145/2457465.2457470

[10] Paul A, Chen B - W, Bharanitharan K, Wang J - F (2013*) "Video search and indexing with reinforcement agent for interactive multimedia services. "* ACM Trans Embed Comput Syst 12 (2): 1–16. doi: 10.1145/2423636.2423643

[11] Kadiyala S, Shiri N (2008) *"A compact multi - resolution index for variable length queries in time series databases. "* Knowl Inf Syst 15 (2): 131–147

[12] Wu K, Shoshani A, StockingerK (2010) *"Analyses ofmulti - level and multi - component compressed bitmap indexes"*. ACM Trans Database Syst 35 (1): 1–52.

[13] Cheng J, Ke Y, Fu AW - C, Yu JX (2011) Fast graph query processing with a low - cost index. VLDB J 20 (4): 521–539

[14] Sebastiani F (2002) *"Machine learning in automated text categorization"*. ACM Comput Surv 34 (1): 1–47. doi: 10.1145/505282.505283