Detecting Text Similarity on a Scalable No-SQL Database Platform

Sergey Butakov
IS Security and Assurance
Concordia University of Edmonton
Edmonton, Canada
sergey.butakov@concordia.ab.ca

Stepan Murzintsev, Aleksandr Tskhai
Mathematics and Applied Informatics
Altai Academy of Economics and Law
Barnaul, Russia
o.l00@yandex.ru, taa1956@mail.ru

***Abstract*—The paper looks at the platform scalability problem for near-to-similar document detection tasks. The application areas for the proposed approach include plagiarism detection and text filtering in data leak prevention systems. The paper reviews limitations of the current solutions based on the relational DBMS and suggests data structure suitable for implementation in no-SQL databases on the highly scalable clustered platforms. The proposed data structure is based on "key-value" model and it does not depend on the shingling method used to encode the text. The proposed model was implemented on the clustered MongoDB platform and tested with the large dataset on the platform that was scaled up horizontally during the experiment. The experiments indicated the applicability of the proposed approach to near-to-similar document detection.**

Keywords— scalable platforms, no-SQL, text search, plagiarism detection, data loss protection

I. INTRODUCTION

Near-to-similar document detection is a common task that surfaces in many applications such as Plagiarism Detection (PD) or text filtering in Data Loss Protection (DLP) systems. Fast text comparison is not a new research area but it has been getting an additional momentum in the last decade along with Big Data concept. The application areas vary from keyword search in typical internet search engines to paraphrased text detection in literature texts [1]. Obviously, the scalability of search algorithms and platforms plays critical role especially for smaller companies that cannot afford high-end infrastructure or for startup companies that provide search services and prefer platform investments that would match business growth. Lack of horizontally scalable algorithms and platforms increases the market entrance costs for the new companies affecting the competition, quality, and variety of text search services.

The main problem that this research is trying to address is the scalability of near-to-similar document detection on the commodity hardware. Majority of suggested algorithms provide good detection ratio and acceptable false positives/negatives but may not be scalable to work with the datasets that include millions of documents or would require expensive infrastructure to handle such datasets [2]. This paper proposes to use scalable infrastructure based on typical commodity hardware. On the software side, the solution utilizes open source no-SQL DBMS.

The rest of the paper is organized as follows: next section briefly describes the problem of horizontal platform scalability for large text databases. The subsequent two sections outline proposed data structure and results of two experiments that confirmed its applicability on the clustered database platform.

II. SCALING UP VARIOUS SOLUTIONS FOR TEXT COMPARISON

The area of fast document comparison has been studied since 1970s. One large class of the algorithms that are trying to address the issue of near-to-similar document detection is based on the text shingling approach. The approach assumes that the plain text is being split into n-grams or shingles. These shingles or some subsets of them [3] [4] are being used for the text comparison. Shingling methods play significant role in the quality of the search, which can be evaluated through Recall, Precision, and Granularity of the detection results [5]. Using subsets of the original shingle set may improve the speed of the search as some algorithms take only as much as 0.02 of the original shingle set into the document fingerprint, which in turn, is being used for the search [4]. Even though in such cases, the document fingerprint will include only 0.02 × [number of characters] shingles the database will scale up quite quickly if dataset includes millions of documents. Experiments provided in [4] with 500,000 documents generated over 38 million fingerprints. Obviously performing search on such large tables requires powerful infrastructure. A number of known solutions for plagiarism detection use Relational Database Management Systems (RDBMS) to store shingles and perform search on them. However, as studies indicate, such approach may suffer from the RDBMS performance limitations [6].

Apart from the utilizing search and storage capacities in or RDBMS, some other attempts have been made to speed up the comparison. For example, use of probabilistic data structure called Bloom filter [7] was suggested as a way to increase the speed of text comparison [8] [9]. The approach provided reasonable results in terms of the speed and quality of the detection but has an obvious applicability limitation as it assumes storing shingles in the RAM. Such a requirement of having large amounts of RAM available for the application may cripple system implementations in cases where the user cannot afford the infrastructure with an excessive amount of RAM.

Approach to use scalable non-SQL based DBMS for keyword indexing in larger documents was proposed in [10]. The authors concentrated on the keyword-based search and proposed combination of the indexing algorithms on the scalable cloud-like infrastructure with multi-threaded indexing. The focus area of the approach is more on the indexing of single keywords similar to a typical internet search engine [10]. Focus area of the research presented in this paper is text similarity, which cannot be translated to the keyword search as similarity analysis looks into combinations of words/n-grams.

The brief review of the current developments in the area of near-to-similar document detection indicated the need for the development of data model that would be relatively easy to implement on a horizontally scalable platform. The main research problem for this project was to develop such a model and test it on a scalable infrastructure. The following two sections outline the description on the proposed text representation in no-SQL database and results of its tests on the scalable clustered platform.

III. 3. Suggested data structure

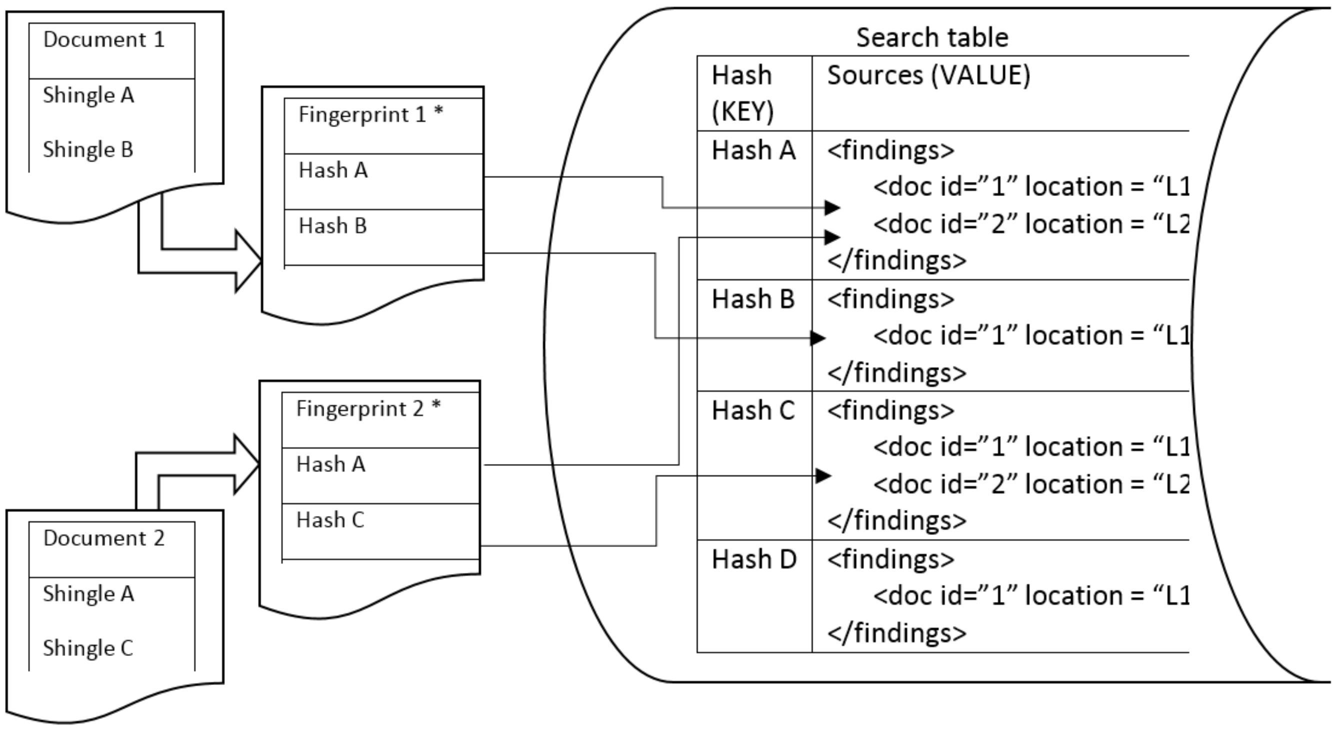
Suggested data structure assumes that some independent algorithm prepares shingles representing the text. Such independence from a shingling algorithm helps to achieve two main goals:

a) Make it possible to perform pre-processing and actual comparison on different platforms thus improving computational efficiency of the platform.

b) Make it possible to use shingling methods that can detect obfuscated texts by producing same sequences of shingles for similar texts.

The most time consuming search operation will be to find the subset of documents that have at least one common shingle with the document that is being filtered. Formally the task is to find a subset $S \subset T$, $S=(s_i)$ where T is set of original documents loaded in the filter; $H(x) = (h_j)$, h_j is the hashing function and for each s_i there is at least one common hash value with d: $|H(s_i) \cap H(d)| \geq 1$. The suggested solution for the scalability problem is to aggregate documents with the same shingles while they are being inserted into the filter. In this case, the data structure can be presented in the "key-value" form making it relatively easy to use it on scalable non-SQL platforms. The suggested structure will look like the following pair: $\{\langle h_k \rangle - \langle S_k=(s_i,\ i=1,n)\rangle\}$ where $S_k \subset S$. In other words, each hash will have attached structured description of the documents it belongs to. Figure 1 provides the example of such representation.

As it can be seen from the formal description of the model and figure 1, the proposed data structure allows to represent a shingle and related document(s) in a flat 2D table which can be



*** To simplify the picture we assume that each shingle will produce a hash that will be included in the fingerprint by shingling algorithm.**

Figure. 1. KEY-VALUE representation for no-SQL database

easily translated into “key-value” structure suitable for implementation in no-SQL environment. The following experiments evaluated the capacities of such a model for practical applications.

IV. Experiments

Experiments in this work have been aimed at the confirmation of the applicability of the distributed no-SQL databases for near to similar document detection. The applicability was evaluated by two criteria: scalability of the solution and its performance on the large set of documents. Scalability of the actual implementation was tested on the clusters with different number of nodes with the requirement not to change the actual source code of the program. Performance of the solution was evaluated using two factors: (1) time required to load documents in the database and (2) time required to perform the search.

For implementation MongoDB was used as one of the most popular no-SQL DBMS with fast growing user base (www.mongodb.org). The distributed infrastructure in MongoDB terms assumes using so called shards, where single shard is essentially a portion of the database nested on one machine. The actual program implementation was found scalable, as it was possible to port the program between MongoDB setup with no shards (just on machine), Mongo DB cluster with three shards, and finally cluster with six shards.

For the load and search times tests the experiments used a dataset comprised of 3,917,453 documents taken from English segment of Wikipedia. Prior to the experiment, the text corpus was cleaned up from the wiki markup tagging.

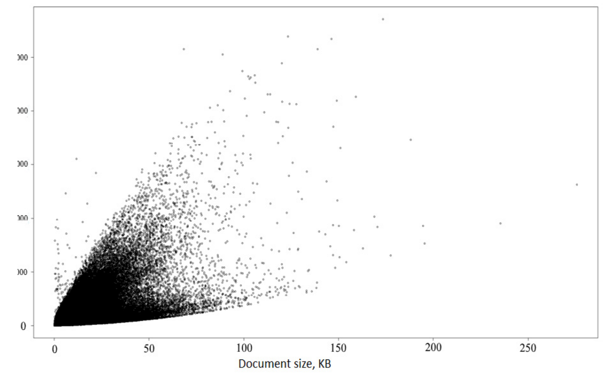
A. Document load time

Although document load is one time procedure that will be performed only once when the filtering database is being populated, the speed of the process has to be evaluated to make sure the load time is acceptable when there are millions of the documents already seat in the storage. The average speed for the data load is presented in Table 1. They deem to be acceptable as load time of 11 ms/KB on six shards can be interpreted as about one second for 6-page document.

TABLE I. AVERAGE LOAD SPEED PER 1 KB OF THE TEXT.

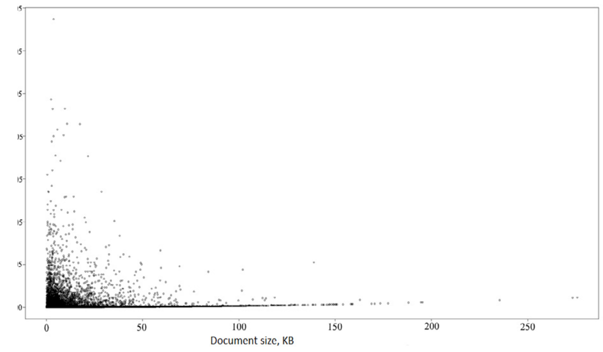
	No shards	**3 shard**	**6 shards**
Avg. load time, ms/KB	87	21	11

The load time is also presented on Figure 2 as a function of the document size. As analysis of these diagrams indicates, the time varies significantly even for the documents of the similar size. The main reason for such variations is the fact that the speed actually depends on the number of documents that already loaded in the table. Figure 3 represents another view on the process. It shows the load per KB of text time depending on the number of preloaded documents. As it can be seen from diagrams on Figure 3 the load speed is near-to-linear and it grows slower with increasing number of nodes (shards) in the cluster. The vertical bars on the graphs indicate short time increases during re-indexing that happens during the load time. This experiment confirms that the proposed text representation model is functional on the scalable infrastructure and that increase on the number of shards improves system performance.



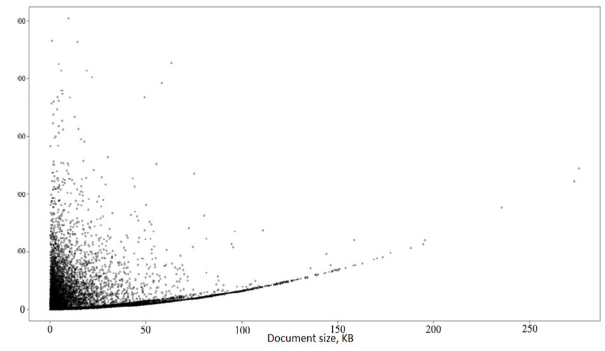
(a)

No shard



(b)

Three shards



(c)

Six shard

B. Exploring the search time.

The second experiment was aimed at the evaluation of the similarity detection speed. Typical similarity detection process includes two steps [5]: preliminary search and detailed markup. Preliminary search assumes selection of the candidate documents from the large database of the potential sources. It helps to reduce the search domain from millions to tens or hundreds of documents. The second steps assumes detailed comparison with the aim to highlight near-to-similar fragments in two texts. The experiment conducted in this research simulated only the preliminary selection as the most computationally extensive process.

The loaded database from the first experiment was used to perform search. One thousand documents randomly selected from the preloaded corpus were used to check the search speed. Obviously each document generated 100% overlap with itself and potentially numerous overlaps with the another document(s) but the purpose of the experiment was to evaluate speed of the detection rather than quality of the search. The latter was not in the focus as shingling algorithms typically provide robust search quality on the non-obfuscated documents.

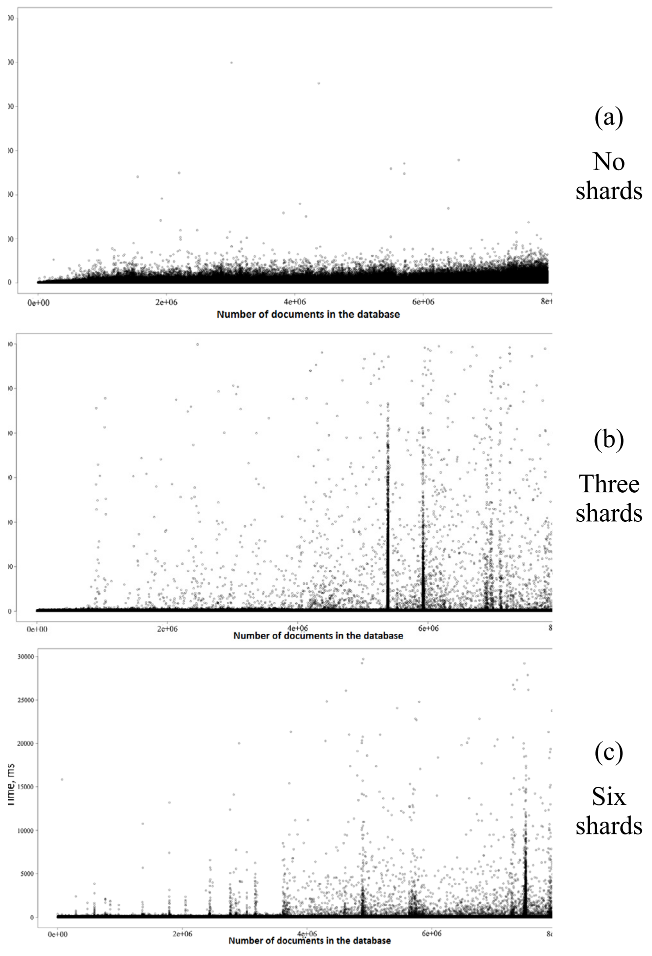


Figure 3. Table size and speed of loading process

Table 2 shows the average search speed which happened to be twice less on the distributed infrastructure if compared with the setup where the entire database is located on one machine. The search times presented on Figure 4 show that on no-sharded infrastructure the search time is not really linear and varies significantly even for the documents of similar size. The search time on the sharded infrastructure is almost linear to the document size which suggests good scalability of the search function on the distributed database. It was also noted that the average search time on six shards was slightly higher if compared to three shards. Such observation requires further research and could be possibly attributed to either the case that infrastructure with six shards is excessive for four million documents or the MongoDB DBMS routing server was not able to cope with the workload distribution job properly.

TABLE II. AVERAGE SEARCH SPEED PER 1KB OF THE TEXT.

	No shards	**3 shard**	**6 shards**
Avg. search time, ms/KB	159.1	77.7	78.6

In summary both experiments confirmed the scalability of the systems and produced acceptable search results that can be used in applications such as document filtering in DLP systems or similarity search in PD systems.

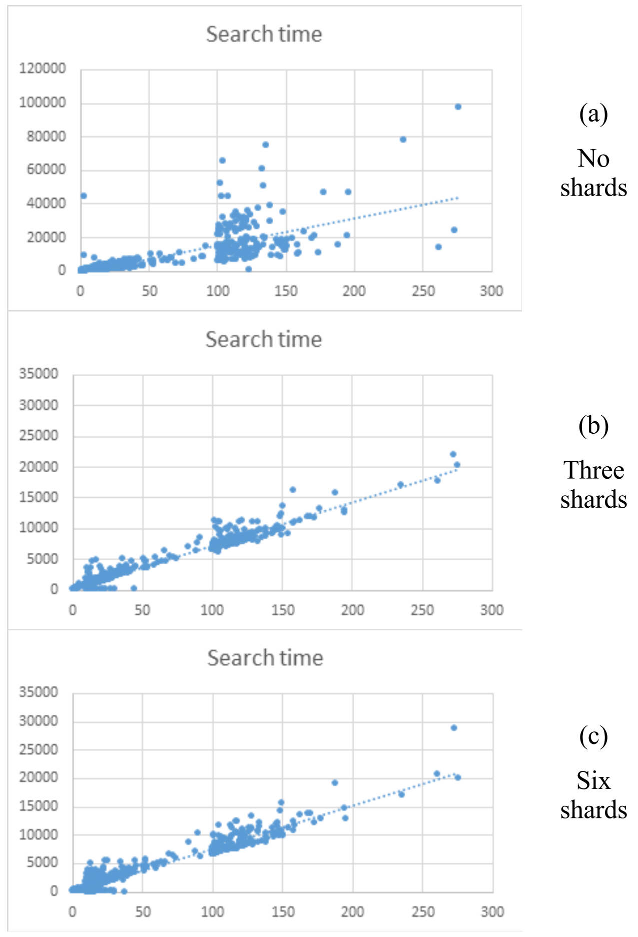


Figure 4. Document size and search time

V. CONCLUSION

The main result of the project is the proposed data structure that allows representing text in the "key-value" form suitable for a typical no-SQL DBMS. Although the tested implementation used Winnowing-like algorithm [4] the proposed model is in fact independent from the shingling algorithms. Therefore using other shingling algorithms that, for example, address the issue of text obfuscation, can be done without changing the database structure.

Quantitative evaluation of the speed of candidate documents selection process indicated acceptable times for practical purposes. Time of about 80ms per KB of text can be translated in one second delay for 6-page document which is acceptable for most of the similarity detection applications. For example, such a performance would allow to process about 85,000 documents with few million documents in 24 hours. This capacity sounds more than sufficient for a typical PD system, which serves for one or more medium size schools. Although these calculations are provided for the candidate documents selection only, the service of detailed comparison may run on completely independent separate infrastructure which will experience much less load. Given the fact that the experiments were performed on the conventional computers, it can be summarized that the proposed model provides acceptable performance on highly scalable and inexpensive platform. Further research might be required to seek the indicators that would flag the necessity to add/remove nodes from the platform that performs text comparison tasks.

REFERENCES

[1] J.-G. Ganascia and A. Del Lungo, "Automatic detection of reuses and citations in literary texts," Lit Linguist Computing, vol. 29, no. 3, pp. 412-421 , 2014

[2] D. Pohuba, T. Dulik and P. Janku, "Automatic evaluation of correctness and originality of source codes," in 10th European Workshop on Microelectronics Education (EWME), 2014.

[3] R. M. Karp and M. O. Rabin, IBM Journal of Research and Development, vol. 31, no. 2, pp. 249-260, 1987.

[4] S. Schleimer, D. S. Wilkerson and A. Aiken, "Winnowing: local algorithms for document fingerprinting.," in Proceedings of the 2003 ACM SIGMOD international conference on Management of data., 2003.

[5] M. Potthast, B. Stein, A. Eiselt, A. Barron-Cedeno and P. Rosso, "Overview of the 1st international competition on plagiarism detection.," in 3rd PAN Workshop. Uncovering Plagiarism, Authorship and Social Software Misuse., 2009.

[6] V. Shcherbinin and S. Butakov, "Using Microsoft SQL server platform for plagiarism detection," in Proc. SEPLN, 2009.

[7] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors.," Communications of the ACM, vol. 13, no. 7, pp. 422-426, 1970.

[8] S. Geravand and M. Ahmadi, "A Novel Adjustable Matrix Bloom Filter-Based Copy Detection System for Digital Libraries," in 2011 IEEE 11th International Conference on Computer and Information Technology (CIT), 2011.

[9] L. Shi, S. Butakov, D. Lindskog, R. Ruhl and E. Storozhenko, "Applicability of Probablistic Data Structures for Filtering Tasks in Data Loss Prevention Systems," in 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Gwangju, 2015.

[10] V. Srikrishnan, E. Sivasankar and R. Pitchiah, "A performance comparison of scheduling distributed mining in cloud," in 2014 First International Conference on Networks & Soft Computing (ICNSC), 2014.