When Similarity Digest Meets Vector Management System: A Survey on Similarity Hash Function

Zhushou Tang[1] Lingyi Tang[2] Keying Tang[3] Ruoying Tang[4]

[1]PWNZEN InfoTech Co., LTD [2]The Webb Schools
[3]Shanghai Fushan Foreign Language School [4]Shanghai Pinghe School

Abstract

The booming vector manage system calls for feasible similarity hash function to perform similarity analysis. In this paper, we make a systematically survey on the existent well-known similarity hash functions to tease out the satisfied ones. We conclude that the similarity hash function $MinHash$, $Nilsimsa$ can be directly plugged into the pipeline of similarity analysis using vector manage system. After that, we make a brief and empirical discussion on the performance, drawbacks of the these functions and highlight $MinHash$ and the variant of $SimHash$ are the best for vector management system for large-scale similarity analysis.

1 Introduction

In this section, we introduce the *vector management system*, *similarity hash function*, and analyze the interleaving between them.

Vector Management System. The proliferation of vector management system (*e.g.*, Milvus [WYG+21], Analyticdb-v [WWW+20], PASE [YLFW20], faiss [JDJ19], Vearch [LLG+18], SPTAG [CWL+18]) satisfys the management of sheer volume of high-dimensional vectors generated by data science and AI applications. By taking advantage of GPUs or CPUs and sorts of optimizations (*e.g.*, indexing), these systems provide versatile similarity functions, including Euclidean, Inner product, Jaccard, Tanimoto, Hamming, SuperStructure, and SubStructure distance measurement, allowing exploring similar vectors in an effective approach.

Similarity Hash Function. The similarity hash function is wildly used in malware triage [NJS19, LSB+15, Ser15, JBV11], forensic analysis [SLC+09], plagiarism detection [SWA03]. As opposed to cryptographic hash function, similarity hash function maps features of an entity (*e.g.*, binary, text) into some high dimensional space, namely digest, since the distance between similarity hashing digests is an approximation of original features, similarity can be measured by distance within these digests.

Gap between the Vector Management System and Similarity Hash Function. At a first glance, the digest generated by a similarity hash function can be directly fed into vector management system for similarity analysis. However, not all digests meet the requirement of vector management system. As depicted by Digest 1, the n at the beginning of the digest generated by ssdeep [Kor06] is used for the first round comparison, and then used to select one of the subsequent two non-fixed length digests for the second round comparison, such logic increase the complexity of comparison, and the variant-length digest is not supported by the vector management system in the mean time.

$$\underbrace{1536}_{\text{n}} : \underbrace{T0tUHZbAzIaFG91Y6pYaK3YKqbaCo/6Pqy45kwUnmJrrevqw + oWluBY5b32TpC0}_{\text{1st chunk}} : \underbrace{T0tU5s7ai6ptg7ZNcqMwUArKvqfZlMC0}_{\text{2nd chunk}} \tag{1}$$

Overlap of the Vector Management System and Similarity Hash Function.

It is observed that there are overlapping functions between the vector management system and similarity hash function. For example, the *Locality-Sensitive Hashing* (LSH) is used by both some similarity hash functions (*e.g.*, SimHash) and vector management system (*e.g.*, indexing of faiss), the overlapping approximation makes the analysis result unpredictable.

The main goal of this work is to make a systematically survey on the existent similarity hash functions and figure out these can be plugged into the vector management system.

2 Survey on the Similarity Hash Functions

This section starts with building new triage for these functions, then dive into each similarity hash function to find the vector management system acceptable ones.

2.1 New triage

In general, the similarity hash function consists two parts: *(i)* digest generation and *(ii)* digest comparison. The digest generation is further separated into feature extraction and feature encoding process. Based on how the digest function encode the features and compute the final digest, the current triage of digest function can be separated into: *Context-Triggered Piecewise Hashing* (CTPH, namely fuzzy hash), *Block-Based Hashing* (BBH), *Locality-Sensitive Hashing* (LSH), *Statistically-Improbable Features* (SIF). *e.g.*, Oliver *et al.* [OCC13] categorized the similarity digest function into three families: CTPH, BBH, and SIF; Li *et al.* [LSB+15] separate them into CTPH and BBH. Besides, Gayoso *et al.* [GMHÁHE14] add one more category *Block-Based Rebuilding* (BBR) to the collection.

Other than the aforementioned categories, we build new triage based on how the digest compared. Considering that the vector management system only accepts/operates fixed-length bit-vector or floating-vector and complex similarity analysis is not supported yet, we separate these functions into three families: *(i) Vector-Hashing* (VH), *(ii) Portable-Vector-Hashing* (PVH) and *(iii) Non-Vector-Hashing* (NVH). The VH family is that the vector can be fed into vector management system directly and exploit the similarity function within the vector management system; PVH family digest contains auxiliary information, but can be tailored to fit the vector management system (driven by extra control); At last, the non-fixed length digest or complex similarity analysis is not supported by vector management system is categorized into NVH family. For example, the ssdeep [Kor06] is categorized as CTPH in conventional, however we classify it as NVH, for the comparison of this function uses weighted edit distance between two digests which is not supported by the vector management system.

2.2 Similarity Hash Function Analysis

To categorize the similarity hash functions, we go through the well-known digest generation schemes, roughly summarize each similarity hash function, and tease out the functions that can be plugged into the pipeline of similarity analysis embedding vector management system. The schemes are listed in Table 1. From this table, we find that MinHash and Nilsimsa belong to VH family, TLSH is the variant Nilsimsa, the auxiliary information appended to digest improves the precision, but does not fit the vector manage system anymore. We also notice that bloom filter complex the comparison, however bucket mapping aligned the digest to fixed-length.

Note that, *(i)* although there are multiple variants for a given scheme, we only discuss the well-known version (*e.g.*, the SimHash [sim21] proposes to use Shingling as a prerequisite for feature extraction, the open source implementation on github uses Jieba instead; Although FKSum [CW08], SimFD [SLC+09], MRSH [RRIM07, BB12]) improve either the efficiency or precision of ssdeep, we take ssdeep nonetheless). Details of these schemes are also summarized by Gayoso *et al.* [GMHÁHE14], we encourage readers to refer this material. *(ii)* Deep Supervised Hashing [LCZ+20] does not discussed here.

3 Discussion

In this section, we discuss the feasible similar hash function, we conclude that industrial adopts to use high efficient function and domain knowledge plays an import role in similarity analysis.

3.1 Tradeoff Between Precision and Performance

Lots of work has evaluated the precision of the similar hash functions [Cha02, BBB12, OFC14, KKC+20], however, the sheer volume of high-dimensional vectors generated by data science and AI applications allow for a certain tolerance on precision. For example, even though the ssdeep is the de facto standard of similarity hash function and integrated into VirusTotal, there is no sign indicate VirusTotal has exploited the digests to find similar binaries. On the other side, the lightweight Simhash is announced to be used by Google for duplicate detection for web crawling [MJDS07], Minhash is used

Scheme	Original Family	Feature Extraction	Feature Generation	Digest Property	Distance Function	New Family	Designed For
MinHash [Bro97]	LSH	Winnowing	Bucket mapping	-	Jaccard Similarity	VH	Binary
dcfldd [NIC02]	BBH	Split the data into sectors or blocks of fixed-length.	Compute the corresponding cryptographic hash value for each of these blocks.	-	-	NVH	Binary
Nilsimsa [DdVPS04]	LSH	Winnowing	Bucket mapping (Pearson hash [Pea90]) followed by encoding the accumulated value to bit-vector under the control of a threshold.	32-bytes length vector	Hamming distance	VH	Text
ssdeep [Kor06]	CTPH	Use Alder-32 (rolling hash) to identify boundaries of a chunk.	Use FNV-hash [Fow91] upon the chunk and use the last 6-bits.	Non-fixed length with auxiliary information, ASCII encoding	Weighted edit distance after the auxiliary information comparison	NVH (variant length digest)	Text
SimHash [SL07]	LSH	Accumulate the occurrence of pre-defined 16 8-bits tags (Shingling).	The combination of sum table	32-bits vector enclosing auxiliary information (*e.g.*, file extension).	Hamming distance	PVH	Text
sdhash [Rou10]	SIF	Get entropy estimates of each 64-bytes block, use winnowing to select the entropy estimates.	Bloom filter (SHA-1)	Multiple bloom filters	Average the maximums of per-filter in one set 'AND' each counterpart filters.	NVH	Text
TLSH [OCC13]	LSH	Winnowing	Bucket mapping (Pearson hash [Pea90]) followed by encoding the accumulated value to bit vector under control of quartile points.	checksum + length + quartile points + encoded value	Circular distance + auxiliary information	VNH	Binary
mvHash-B [BABB13]	BBH	Majority vote followed by RLE (length encoding algorithm)	Bloom filter	Multiple bloom filters	Average the minimums of per-filter in one set 'XOR' each counterpart filters.	NVH	Binary

Table 1: Break down of similarity hash functions.

for Google News personalization [DDGR07], Jang *et al.* [JBV11] also stat that *feature hashing* which maps each feature into exactly one bit in the bit array works well in large-scale malware similarity analysis. Such lightweight scheme brings extra bonus for analysis task, for example, the *feature hashing* solution can pinpoint the exactly the commonality and difference between malware samples in the mean time.

3.2 Role of Domain Knowledge

Although there are literals argue the universality of similarity hash functions [GMHÁHE14], we believe domain knowledge (*a.k.a.* semantic) plays an important role in similarity analysis. Taking android third-party library correlation as an example, different arguments of compilation will dramatically change the binary, leading to similarity detection failure, but works [MWGC16, TXM+19, FR19] take in domain knowledge (*e.g.*, System Call, Abstract Syntax Tree) as feature outperform those without domain knowledge.

References

[BABB13] Frank Breitinger, Knut Petter Astebøl, Harald Baier, and Christoph Busch. mvhash-b-a new approach for similarity preserving hashing. In *2013 Seventh International Conference on IT Security Incident Management and IT Forensics*, pages 33–44. IEEE, 2013.

[BB12] Frank Breitinger and Harald Baier. Similarity preserving hashing: Eligible properties and a new algorithm mrsh-v2. In *International conference on digital forensics and cyber crime*, pages 167–182. Springer, 2012.

[BBB12] Frank Breitinger, Harald Baier, and Jesse Beckingham. Security and implementation analysis of the similarity digest sdhash. In *First international baltic conference on network security & forensics (nesefo)*, 2012.

[Bro97] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[Cha02] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[CW08] Long Chen and Guoyin Wang. An efficient piecewise hashing method for computer forensics. In *First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008)*, pages 635–638. IEEE, 2008.

[CWL+18] Qi Chen, Haidong Wang, Mingqin Li, Gang Ren, Scarlett Li, Jeffery Zhu, Jason Li, Chuanjie Liu, Lintao Zhang, and Jingdong Wang. Sptag: A library for fast approximate nearest neighbor search, 2018.

[DDGR07] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280, 2007.

[DdVPS04] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. An open digest-based technique for spam detection. *ISCA PDCS*, 2004:559–564, 2004.

[Fow91] G Fowler. Fowler/noll/vo (fnv) hash. *http://isthe.com/chongo/tech/comp/fnv*, 1991.

[FR19] Johannes Feichtner and Christof Rabensteiner. Obfuscation-resilient code recognition in android apps. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–10, 2019.

[GMHÁHE14] Víctor Gayoso Martínez, Fernando Hernández Álvarez, and Luis Hernández Encinas. State of the art in similarity preserving hashing functions. 2014.

[JBV11] Jiyong Jang, David Brumley, and Shobha Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 309–320, 2011.

[JDJ19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[KKC+20] Dongkwan Kim, Eunsoo Kim, Sang Kil Cha, Sooel Son, and Yongdae Kim. Revisiting binary code similarity analysis using interpretable feature engineering and lessons learned. *arXiv preprint arXiv:2011.10749*, 2020.

[Kor06] Jesse Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital investigation*, 3:91–97, 2006.

[LCZ+20] Xiao Luo, Chong Chen, Huasong Zhong, Hao Zhang, Minghua Deng, Jianqiang Huang, and Xiansheng Hua. A survey on deep hashing methods. *arXiv preprint arXiv:2003.03369*, 2020.

[LLG+18] Jie Li, Haifeng Liu, Chuanghua Gui, Jianyu Chen, Zhenyuan Ni, Ning Wang, and Yuan Chen. The design and implementation of a real time visual search system on jd e-commerce platform. In *Proceedings of the 19th International Middleware Conference Industry*, pages 9–16, 2018.

[LSB+15] Yuping Li, Sathya Chandran Sundaramurthy, Alexandru G Bardas, Xinming Ou, Doina Caragea, Xin Hu, and Jiyong Jang. Experimental study of fuzzy hashing in malware clustering analysis. In *8th Workshop on Cyber Security Experimentation and Test ({CSET} 15)*, 2015.

[MJDS07] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, pages 141–150, 2007.

[MWGC16] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen. Libradar: fast and accurate detection of third-party libraries in android apps. In *Proceedings of the 38th international conference on software engineering companion*, pages 653–656, 2016.

[NIC02] H NICHOLAS. Dcfldd. *http://dcfldd.sourceforge.net/*, 2002.

[NJS19] Nitin Naik, Paul Jenkins, and Nick Savage. A ransomware detection method using fuzzy hashing for mitigating the risk of occlusion of information systems. In *2019 International Symposium on Systems Engineering (ISSE)*, pages 1–6. IEEE, 2019.

[OCC13] Jonathan Oliver, Chun Cheng, and Yanggui Chen. Tlsh–a locality sensitive hash. In *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, pages 7–13. IEEE, 2013.

[OFC14] Jonathan Oliver, Scott Forman, and Chun Cheng. Using randomization to attack similarity digests. In *International Conference on Applications and Techniques in Information Security*, pages 199–210. Springer, 2014.

[Pea90] Peter K Pearson. Fast hashing of variable-length text strings. *Communications of the ACM*, 33(6):677–680, 1990.

[Rou10] Vassil Roussev. Data fingerprinting with similarity digests. In *IFIP International Conference on Digital Forensics*, pages 207–226. Springer, 2010.

[RRIM07] Vassil Roussev, Golden G Richard III, and Lodovico Marziale. Multi-resolution similarity hashing. *digital investigation*, 4:105–113, 2007.

[Ser15] Shadow Server. Fuzzy clarity: Using fuzzy hashing techniques to identify malicious code, 2015.

[sim21] Hash functions. https://github.com/yanyiwu/simhash/blob/master/README_EN.md, September 2021. Accessed September 1, 2021.

[SL07] Caitlin Sadowski and Greg Levin. Simhash: Hash-based similarity detection. *Technical report, Google*, 2007.

[SLC+09] Kimin Seo, Kyungsoo Lim, Jaemin Choi, Kisik Chang, and Sangjin Lee. Detecting similar files based on hash and statistical analysis for digital forensic investigation. In *2009 2nd International Conference on Computer Science and Its Applications, CSA 2009*, page 5404198, 2009.

[SWA03] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85, 2003.

[TXM+19] Zhushou Tang, Minhui Xue, Guozhu Meng, Chengguo Ying, Yugeng Liu, Jianan He, Haojin Zhu, and Yang Liu. Securing android applications via edge assistant third-party library detection. *Computers & Security*, 80:257–272, 2019.

[WWW+20] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. *Proceedings of the VLDB Endowment*, 13(12):3152–3165, 2020.

[WYG+21] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021.

[YLFW20] Wen Yang, Tao Li, Gai Fang, and Hong Wei. Pase: Postgresql ultra-high-dimensional approximate nearest neighbor search extension. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2241–2253, 2020.