On Tradeoffs Between Document Signature Methods for a Legal Due Diligence Corpus

Adam Roegiest, Edward Lee
Kira Systems
{adam.roegiest,edward.lee}@kirasystems.com

ABSTRACT

While document signatures are a well established tool in IR, they have primarily been investigated in the context of web documents. Legal due diligence documents, by their nature, have more similar structure and language than we may expect out of standard web collections. Moreover, many due diligence systems strive to facilitate real-time interactions and so time from document ingestion to availability should be minimal. Such constraints further limit the possible solution space when identifying near duplicate documents. We present an examination of the tradeoffs that document signature methods face in the due diligence domain. In particular, we quantify the trade-off between signature length, time to compute, number of hash collisions, and number of nearest neighbours for a 90,000 document due diligence corpus.

ACM Reference Format:
Adam Roegiest, Edward Lee. 2019. On Tradeoffs Between Document Signature Methods for a Legal Due Diligence Corpus. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331311

1 INTRODUCTION

In legal due diligence, lawyers are tasked with reviewing a target company's contracts and agreements to identify any potential risk that may result from a merger or acquisition. Often, the deadlines for review are tight and senior lawyers can be expected to provide an assessment on as much information as could be gathered in the time frame. While automated methods [17] can aid in the identification of relevant clauses, they still need to be reviewed and summarized. From a project management view, there is a desire to assign work to the "best" reviewer for a document set while also minimizing the amount of review to be performed by identifying near duplicates. However, near duplicates can span a wide range of categories in due diligence from signed and unsigned versions of a contract to documents based on the same form to more traditional near duplicates (e.g., small changes between documents).

Accordingly, in this work, we investigate the tradeoffs in various near-duplicate detection algorithms, specifically those relying on document signatures. We consider document signatures since they are relatively quick to compute and can be made to efficiently retrieve near-duplicates [8, 13, 16]. The efficiency of the used mechanism is of paramount importance due to the project-based nature of due diligence work. Users wish to have a system ingest documents and have them ready to use as soon as possible to ensure that they can provide a timely and accurate report. Accordingly, any mechanism used to identify near-duplicates has an implicit requirement to be as close to real-time as possible. Delays would mean that project management is delayed and resources may be misappropriated. Moreover, the mechanism used should be of high enough accuracy to ensure that the number of false identifications is minimized and documents can be assigned to the appropriate reviewer.

In this paper, we investigate three document signature techniques, based on random vector projections, for their applicability to near-duplicate detection in a due diligence context. We examine Minhash [1], Simhash [4], weighted and unweighted TopSig [11] methods on a range of signature lengths from 16 to 1024 bits. While Simhash and TopSig are related techniques, we use them to investigate different formulations of the underlying random projection methods (Section 3). Using a 90,000 document sample of the EDGAR repository,[1] we examine the time taken to produce document signatures, the number of unique signatures, and the number of collisions per signature for each technique. These provide baseline measures of the tradeoffs that we might expect to happen in user projects. Collisions are not our sole measure since users may wish to tolerate more variance when identifying near duplicates (e.g., documents based on the same form). Accordingly, we examine the distributions of nearest neighbours when signatures differ by 1 and 3 bits. While more restrictive than has been done in web contexts [13], we do not seek to be too permissive as these documents by their nature have more similar content and structure than two arbitrary web pages might. Moreover, some preliminary internal user testing has shown that being too permissive degrades the user experience.

2 RELATED WORK

Document signatures have a long history in Information Retrieval research [10] and we make no attempt to provide a summary of all available resources. Instead, we aim to highlight why particular approaches may not be appealing in our domain and use case. The *I Match* algorithm of Chowdhoury et al. [5] and the refinement by Kolcz et al. [15] uses *IDF* to select representative features for inclusion in document signatures. Such approaches will require collection dependent tuning and may not perform well on projects of varying sizes. For similar reasons, we do not find proposals [19–21] to learn optimal hash functions as lucrative since the learning process may delay the ability of project managers to immediately

[1] https://www.sec.gov/edgar.shtml

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331311

leverage near-duplicate information (i.e., waiting for the hashes to be learned) and cross-project models may provide bad encodings (e.g., if two projects are very different).

While some work into curating collections for use in evaluating near duplication detection exists [5–7, 18], such corpora have primarily focused on document collections that do not reflect our problem domain. Finally, we are not concerned in this work with retrieval of documents using their signatures and leave investigating the applicability such methods [2, 3, 12, 14] to future work.

3 EXPERIMENTAL DESIGN

To conduct our experiments, we collected approximately 90,000 documents from the publicly accessible EDGAR repository of filings that the United States Securities and Exchange Commission requires publicly traded companies to submit. We collected documents from January 2017 until March 2018 that correspond to various agreements, articles of incorporation, and material contracts (e.g., leases, supply agreements).[2] Such documents form a reasonable assortment of document types that could appear in a due diligence document collection. According to in-house experts, we might reasonably expect between 3%-10% of near-duplicates in EDGAR depending on one's exact definition (e.g., whether documents based on the same form document are "near duplicates"). Accordingly, we might expect a good document signature method would have no more than ~9,000 signature collisions in total.

To maintain experimental consistency between the different document signatures, each document in the collection was featurized into sliding windows of UTF-8 character 4-grams. Moreover, as capitalization and punctuation can play an important aspect in many legal documents (e.g., acronyms), we did not perform any transformations to the source text (e.g., lowercasing, stopword removal, etc). Using these features, we then computed document signatures according to the MinHash [1], Simhash [4], and TopSig [11] algorithms for signatures lengths in the set of $N \in \{32, 64, 128, 256, 512, 1024\}$ bits. The details of our implementations follow and code for these methods will be released publicly.[3]

For TopSig, we associated each feature with a sparse random vector $\{-1, 0, 1\}^N$ such that i'th entry had approximately a $\sqrt{N}^{-1}$ chance of generating a non-zero entry. For a given document, we computed unweighted and weighted versions of TopSig by taking the sum of each feature's random vector and the sum of each feature's weighted random vector using the formula of Geva and De Vries [11], respectively. The final N-bit vector is generated by setting the i'th bit to 1 if the summed vector's i'th entry is positive. Random vectors were stored in an SQLite database stored in RAM, indexed by their corresponding feature to provide convenient look-up across multiple runs.

Simhash is essentially a more widely known predecessor to TopSig that has used smaller bit spaces [9]. To implement Simhash, we compute the SHA-3[4] hash of each feature and run the unweighted TopSig algorithm. We used SHA-3 as it produces sufficiently random and sparse vectors while being reasonably efficient.

[2]More specifically, Exhibits 1, 2, 3, 4, and 10.
[3]See https://github.com/kirasystems/science.
[4]As it allows generating arbitrary length hashes.

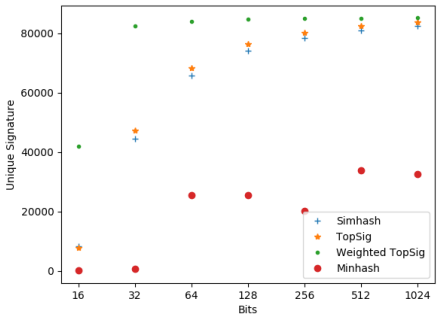


Figure 1: The number of unique document signatures generated for each method at varying bit lengths.

For Minhash, MD5, SHA1, SHA-256, and SHA-512 hashes of every feature were generated for the same reasons as Simhash. For each document, we generated an N-bit MinHash in the following way: we truncate each feature hash to the first $\frac{N}{4}$ bits, select the minimum hash value for each hashing function, and concatenate the truncated minimum hashes together to form a single Minhash. While this is a simpler implementation than is used in the literature [6, 8, 13], we intended this as a proof of concept to determine whether a more complex solution would be worth the effort.

4 SIGNATURE PROPERTIES

In examining the number of unique document signatures for each method (Figure 1), we see that the weighted TopSig vastly outperforms all other methods for small N. In general, we might expect this since the weighted version will take into account the relative importance of various terms, whereas unweighted TopSig and Simhash treat everything equally. For larger N, all three random projection methods end up converging on similar numbers of unique signatures which is not surprising since features may then be sufficiently distributed to ensure that there are less coincidental collisions. It is worth noting that it appears that randomly generating the projection vectors yields less collision prone signatures than a cryptographic hash. Moreover, initial experiments with Simhash and SHA-512 yielded less effective signatures which indicates that choice of hash is also important. Further investigation is needed to determine if we could combine corpus statistics and SHA-3 hashes to produce equivalent results to weighted TopSig.

Our simple Minhash implementation appears to catastrophically fail and does not exceed more than 35,000 unique document signatures. As we have noted, our implementation is much less complex than in the literature[6, 8, 13] as it was intended to be a proof of concept to determine whether additional complexity would be worth the effort. Based on these results, we do not believe that a more standard implementation would substantially outperform the other methods tested and so report only on the simple implementation.

Table 1 reports the average time taken in seconds to produce document signatures using each method. Simhash appears to be generally much quicker than either TopSig variant. This is not surprising since it doesn't require reading from any outside sources and can just compute the document signature using only the features, whereas TopSig must first get the random vectors to perform

Bits	Simhash	TopSig	Weighted TopSig	Minhash
16	0.166 (0.611)	0.572 (0.546)	0.577 (0.549)	0.832 (0.689)
32	0.145 (0.370)	0.587 (0.558)	0.598 (0.567)	0.831 (0.689)
64	0.165 (0.431)	0.585 (0.553)	0.595 (0.561)	0.834 (0.692)
128	0.181 (0.451)	0.582 (0.549)	0.592 (0.556)	0.842 (0.698)
256	0.250 (0.625)	0.573 (0.539)	0.583 (0.546)	0.856 (0.709)
512	0.385 (0.970)	0.557 (0.525)	0.568 (0.531)	0.886 (0.734)
1024	0.610 (1.514)	0.571 (0.536)	0.583 (0.544)	0.929 (0.768)

Table 1: Mean and standard deviation in seconds taken to produce document signatures for each method at each bit length.

Bits	Simhash	TopSig	Weighted TopSig	Minhash
16	11.537 (76.227)	12.378 (86.290)	2.285 (17.438)	1115.035 (9742.314)
32	2.157 (19.315)	2.024 (14.562)	1.163 (10.381)	153.184 (3557.218)
64	1.458 (14.783)	1.403 (11.504)	1.140 (10.268)	3.773 (23.263)
128	1.293 (13.042)	1.254 (10.803)	1.133 (10.233)	3.778 (32.796)
256	1.223 (12.423)	1.195 (10.524)	1.129 (10.215)	4.732 (56.720)
512	1.185 (11.874)	1.162 (10.369)	1.127 (10.208)	2.826 (19.129)
1024	1.162 (11.595)	1.146 (10.294)	1.126 (10.201)	2.943 (18.781)

Table 2: Means and standard deviation in number of collisions per document signature for each method and bit length.

the projection. Though it worth noting that as the number of bits increases, the approaches reach parity in terms of speed which is expected due to the increasing complexity in producing the cryptographic hash. Additionally, TopSig times do not account for random vector creation or corpus statistic collection as these can be done prior to generation of document signatures. Accordingly, TopSig does have some unaccounted for overhead but we do not find it to be prohibitive. On the other hand, TopSig and Minhash do produce fairly consistent times for all bit lengths, owing to the fact that neither needs to generate increasingly longer hashes during generation time. However, Minhash is consistently the slowest despite being so simple. This is further evidence that a more complex Minhash solution may not be the most viable option.

Finally, one might reasonably wonder how many collisions occur on average for each method (Table 2). Across all levels of N TopSig and Simhash perform similarly on average with Simhash tending to have more variance, except at 16 bits. The why behind the difference at 16 bits is not clear and may just be due to SHA-3 producing better random projections. The equivalent behaviour of unweighted TopSig and Simhash comes down to the fact that they are essentially performing the same computation but with vectors of seemingly different quality. Weighted TopSig, as Figure 1 would suggest, vastly outperforms all other methods at 16 and 32 bits. This indicates that Weighted TopSig at 32 bits may provide a compelling implementation choice across all architectures, while tailoring for 64-bit systems would give even better performance than any other method. As we would expect, Minhash does not perform well at any level and has far too many collisions to be useful.

5 NEAREST NEIGHBOURS ANALYSIS

In this section, we explore the distribution of neighbours that each document would have under Simhash and TopSig algorithms for two levels of dissimilarity (1 and 3 bit differences). We omit Minhash as it would not provide sufficient granularity given the results in Table 2. We also only consider three bit lengths (32, 128, 1024) to provide a concise examination across varying levels of refinement. Figure 2 depicts the number of documents which have neighbours within 1 Hamming distance. As we can see, there are 1000s of documents with more than 10 nearest neighbours regardless of the bit length or algorithm chosen. We note that part of this is due to the fact that there are a non-trivial number of documents that did not have any renderable text and so resulted in a trivial document signature (i.e., 0). Though there are still many documents with valid signatures that have many more than 10 neighbours.

Consistent with expectations, Simhash and unweighted TopSig perform similarly at all bit lengths. While there is some minor differences in the 32 bit plot, we are uncertain that these differences are substantive enough to be perceived by users. On the other hand, both approaches appear to be substantially inferior to weighted TopSig but that may be expected at this point. Furthermore, as seen in Figure 1, the refinement of weighted TopSig is less dramatic as we increase bit length. This is not bad but indicates that there are likely diminishing returns on larger lengths for Weighted TopSig. For example, the number of documents with a single neighbour drops from ~15k at 32 bits to ~10k at 128 bits but only drops to ~9k at 1024 bits. While the differences are still substantive, it is unclear how much larger signatures would help.

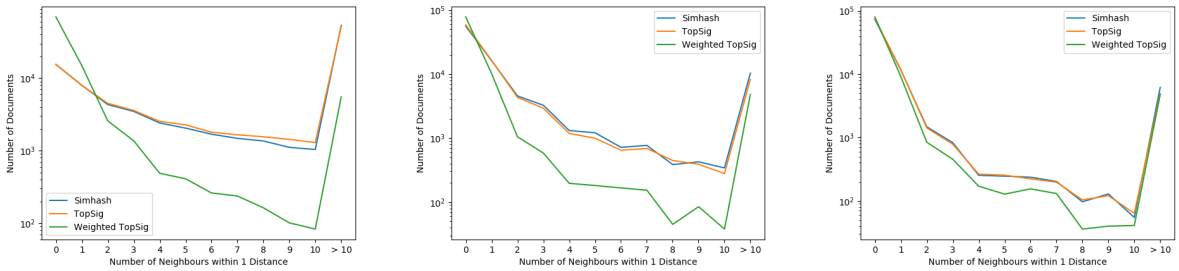
Figure 3 depicts the case when we are more permissive and allow more potential neighbours by considering up to a Hamming distance of 3 for 1024 bit signatures.[5] Simhash and unweighted TopSig still perform equivalently and also find more neighbours than in Figure 2. In contrast, the differences for weighted TopSig are not nearly as large. For example, Simhash and unweighted TopSig "lose" ~4000 documents for the 0 neighbour group but weighted TopSig "loses" only ~400. Of course, this trend is only apparent at 1024 bits. There is much more "loss" of documents from the 0 neighbour group with 32 bit signatures between the 1 and 3 bit difference cases. However, there is still much less "loss" than with the other methods. Accordingly, it would appear to be the case that weighted TopSig is able to better distribute documents in the different signature spaces but is less able to do so effectively for small lengths due to the reduced "space" for variations to take place in. For example, weighted TopSig may be better able to separate different flavours of similar agreements (e.g., agreements with Coca-Cola versus agreements with Nabisco) with 1024 bits than with 32. Further controlled investigation is necessary to examine how such documents get distributed in the various signature spaces.

6 LIMITATIONS AND FUTURE WORK

The most obvious limitation of this work is that we have not used what may be considered an optimal implementation of Minhash. As we have previously stated, this choice was motivated by the necessity of our own internal restrictions and the belief that if a simple implementation did not show promise that a more complex implementation is unlikely to be of sufficient benefit to allocate such effort. Especially since other studies [13, 16] have shown Simhash/TopSig to be efficacious for near duplicate detection tasks.

In addition, none of our implementations are highly tuned for throughput or minimizing memory footprint. Such optimizations

[5] The 32 and 128 plots show the same trends as in Figure 2.



(a) 32 bits (b) 128 bits (c) 1024 bits

Figure 2: Count of documents with set numbers of neighbouring documents within 1 Hamming distance using 32, 128, and 1024 bit signatures.

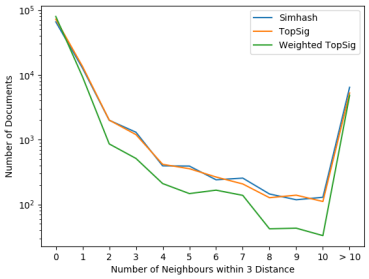


Figure 3: Count of documents with set numbers of neighbouring documents within 3 Hamming distance at 1024 bit document signatures.

can always be performed later to improve the chosen method at a particular bit length. Premature optimization of all algorithms for all bit lengths was seen to be a poor allocation of resources when straightforward implementations that allow reasonably fair comparisons are much easier to produce.

While we have seen that Weighted TopSig is a front runner in speed and effectiveness, one major limitation of the algorithm is the need for collection statistics. As due diligence work is project based, further work is necessary to examine how many documents are necessary for Weighted TopSig to perform as well as we have observed here. Moreover, additional work is necessary to determine how well we could adapt the algorithm for an online setting (i.e., generating Weighted TopSig signatures as we process documents) rather than waiting to perform generation in batches.

7 CONCLUSIONS

We have presented an examination of several document signature methods on their ability to identify near-duplicates in a legal due diligence corpus. We found that methods based on similar random projection techniques perform well and that taking collection statistics into account can improve the generation of high fidelity document signatures in small dimensions (e.g., 16 and 32 bits). The employment of collection statistics appears to more evenly distribute documents in the signature space and results in fewer nearest neighbours being identified for small changes in bit signatures. We have also seen that using cryptographic hashes rather than projections can produce equivalent document signatures, which could result in simpler implementations.

REFERENCES

[1] A. Broder. 1997. On the Resemblance and Containment of Documents. In *Proc. SEQUENCES '97.*
[2] Timothy Chappell, Shlomo Geva, Anthony Nguyen, and Guido Zuccon. 2013. Efficient Top-k Retrieval with Signatures. In *Proc. ADCS '13.*
[3] Timothy Chappell, Shlomo Geva, and Guido Zuccon. 2015. Approximate nearest-neighbour search with inverted signature slice lists. In *Proc. ECIR '15.*
[4] Moses S. Charikar. 2002. Similarity Estimation Techniques from Rounding Algorithms. In *Proc. STOC '02.*
[5] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe. [n. d.]. Collection Statistics for Fast Duplicate Document Detection. *ACM Trans. Inf. Syst.* 20, 2 ([n. d.]).
[6] Jack G. Conrad, Xi S. Guo, and Cindy P. Schriber. 2003. Online Duplicate Document Detection: Signature Reliability in a Dynamic Retrieval Environment. In *Proc. CIKM '03.*
[7] Jack G. Conrad and Cindy P. Schriber. 2004. Constructing a Text Corpus for Inexact Duplicate Detection. In *Proc. SIGIR '04.*
[8] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google News Personalization: Scalable Online Collaborative Filtering. In *Proc. WWW '07.*
[9] Christopher M. De Vries and Shlomo Geva. 2012. Pairwise Similarity of TopSig Document Signatures. In *Proc. ADCS '12.*
[10] C. Faloutsos. 1990. Signature-based Text Retrieval Methods: A Survey. *Data Eng.* 13, 1 (1990).
[11] Shlomo Geva and Christopher M. De Vries. 2011. TOPSIG: Topology Preserving Document Signatures. In *Proc. CIKM '11.*
[12] Bob Goodwin, Michael Hopcroft, Dan Luu, Alex Clemmer, Mihaela Curmei, Sameh Elnikety, and Yuxiong He. 2017. BitFunnel: Revisiting Signatures for Search. In *Proc. SIGIR '17.*
[13] Monika Henzinger. 2006. Finding Near-duplicate Web Pages: A Large-scale Evaluation of Algorithms. In *Proc. SIGIR '06.*
[14] A. Kent, R. Sacks-Davis, and K. Ramamohanarao. 1990. A signature file scheme based on multiple organizations for indexing very large text databases. *J. Amer. Soc. Info. Sci.* 41, 7 (1990).
[15] Aleksander Kolcz, Abdur Chowdhury, and Joshua Alspector. 2004. Improved Robustness of Signature-based Near-replica Detection via Lexicon Randomization. In *Proc. KDD '04.*
[16] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting Near-duplicates for Web Crawling. In *Proc. WWW '07.*
[17] Adam Roegiest, Alexander K. Hudek, and Anne McNulty. 2018. A Dataset and an Examination of Identifying Passages for Due Diligence. In *Proc. SIGIR 2018.*
[18] Enrique Vallés and Paolo Rosso. 2011. Detection of Near-duplicate User Generated Contents: The SMS Spam Collection. In *Proc. SMUC '11.*
[19] Qifan Wang, Bin Shen, Zhiwei Zhang, and Luo Si. 2014. Sparse Semantic Hashing for Efficient Large Scale Similarity Search. In *Proc. CIKM '14.*
[20] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral Hashing. In *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (Eds.).
[21] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. 2010. Self-taught Hashing for Fast Similarity Search. In *Proc. SIGIR 2010.*