Near Duplicate Detection in an Academic Digital Library

Kyle Williams[‡], C. Lee Giles[†‡]
[‡]Information Sciences and Technology, [†]Computer Science and Engineering
Pennsylvania State University, University Park, PA 16802, USA
kwilliams@psu.edu, giles@ist.psu.edu

ABSTRACT

The detection and potential removal of duplicates is desirable for a number of reasons, such as to reduce the need for unnecessary storage and computation, and to provide users with uncluttered search results. This paper describes an investigation into the application of scalable *simhash* and *shingle* state of the art duplicate detection algorithms for detecting near duplicate documents in the CiteSeer$^{\chi}$ digital library. We empirically explored the duplicate detection methods and evaluated their performance and application to academic documents and identified good parameters for the algorithms. We also analyzed the types of near duplicates identified by each algorithm. The highest F-scores achieved were 0.91 and 0.99 for the simhash and shingle-based methods respectively. The shingle-based method also identified a larger variety of duplicate types than the simhash-based method.

Categories and Subject Descriptors

I.7.5 [**Document and Text Processing**]: Document Capture—*Document Analysis*

General Terms

Experimentation, Measurement, Performance

Keywords

Near duplicate detection, simhash, shingles

1. INTRODUCTION

Digital documents have literally changed the way in which documents are discovered, shared and managed through easy versioning, copying and dissemination. As a result, there has been an explosion in the amount of digital documents that are available and digital libraries have arisen as a means of managing these vast quantities of information. Some digital libraries, such as the arXiv[1], allow for users to submit academic papers for inclusion, whereas others, such as CiteSeer$^{\chi}$[2], automatically collect papers through focused crawling. In both cases, it is possible that near duplicate documents are added to the digital library collections. For instance, in the case of the arXiv, users might make minor revisions to a document and submit it as a new document rather than updating their existing submission. Similarly, in the case of CiteSeer$^{\chi}$, similar versions of a paper may exist at multiple locations on the Web and these multiple versions may be automatically added to the collection as a result of the automatic crawling and ingesting.

There has been significant research in near duplicates on the Web; however, there has not been as much research in detecting near duplicates in digital libraries of academic papers and whether methods for duplicate detection on the Web are easily transferable to this domain. Two state of the art duplicate detection algorithms exist: *simhash* [2] and *shingle*-based methods [1]. In this paper, we investigate the use of these two algorithms for finding near duplicates in CiteSeer$^{\chi}$, which is a real-world digital library of academic papers. We measure the precision and the recall of the two algorithms under varying conditions, we experiment to find suitable parameters for the algorithms, and we investigate the types of duplicates detected by each algorithm.

In presenting these contributions, the rest of this paper is laid out as follows. Section 2 discusses related work and Section 3 describes the duplicate detection algorithms used in this study. Section 4 presents the evaluation and, lastly, conclusions are presented in Section 5.

2. RELATED WORK

A state of the art method for detecting duplicate Web pages was proposed by Broder et al. [1]. Broder et al. made use of shingles for duplicate detection in the AltaVista search engine and described efficient algorithms for finding near duplicates in large collections. The simhash algorithm [2] is another state of the art algorithm for duplicate detection that maps a high dimensional feature space to a fixed-size fingerprint [6] and Manku et al. [6] developed an efficient algorithm for finding duplicate documents in a collection. A study comparing the shingle and simhash methods on a dataset containing over 1.6 billion web pages found that both algorithms worked poorly for detecting duplicate web pages from the same site, but worked well for detecting duplicate

[1]http://arxiv.org/
[2]http://citeseerx.ist.psu.edu/

DocEng'13, September 10–13, 2013, Florence, Italy.

ACM 978-1-4503-1789-4/13/09 ...$15.00.
http://dx.doi.org/10.1145/2494266.2494312.

web pages from different sites [4]. Furthermore, it was found that combining the approaches improved results [4].

A technique for duplicate document detection, known as I-Match, is based on collection statistics with the idea being that removing terms that occur very frequently or very infrequently in a collection is a good basis for identifying duplicates by calculating the checksum of the most significant terms in a document [3]. To detect near duplicate books, techniques have been developed based on the hashing of the metadata associated with each book [7] as well as identifying unique words that appear in books and finding near duplicates by aligning the longest common sequences of words [8]. Lastly, one of the few studies to focus on academic documents used concept trees to detect similar documents [5].

As this discussion has shown, there have been several different approaches to duplicate detection, with the simhash and shingle-based methods being state of the art. In this study, we test the use of these state of the art methods for detecting near duplicate academic documents, with a specific focus on parameters for near duplicate detection and the type of near duplicates detected by each method.

3. ALGORITHMS

3.1 Simhash

The simhash algorithm maps a high dimensional feature space to a fixed-size fingerprint [6]. The process involves calculating a *hash* that represents each document and then detecting near duplicates by identifying documents that have similar hashes. The calculation of the hash is not described here due to space constraints; however, the method is the same as used by Manku et al. [6] with each document being represented by a 64-bit hash and with each token in a document contributing an equal weight to the final bit-hash. The distance (and thus similarity) between document bit-hashes is calculated using the Hamming distance.

To find near duplicates, the method proposed by Manku et al. [6] is used. In this approach, two documents are considered as being near duplicates if the Hamming distance between their two hashes is at most k. For a pre-determined k, the method partitions each document bit-hash into $k+1$ sub-hashes and stores each sub-hash and the ids of documents that contain the sub-hash in $k+1$ tables. For a query document, the hash of the document is calculated and partitioned into $k+1$ sub-hashes. Each of the sub-hashes are looked up in the $k+1$ tables and the Hamming distance is calculated between the full hash of the query document and the full hash of each document in the tables that shares a sub-hash with the query document. Using this approach, if two documents differ by k bits then at least one of the $k+1$ sub-hashes is guaranteed to match.

To find near duplicates, two passes are made through the data. In the first pass, hashes are calculated for all documents and the sub-hashes stored in tables and in the second pass each hash is used as a query while making sure not to match a query document with its entry in the hash tables.

3.2 Shingles

Shingles are sequences of tokens of length w that appear in a document and the similarity of two documents can be calculated based on the number of shingles that they have in common [1]. Since it is computationally infeasible to calculate the similarity of the sets of all of the shingles for every document, a method based on the *sketch* of a document is used instead. To calculate the sketch of a document, each shingle in a document is hashed using h hash functions and a list is maintained of the minimum hash values found for each hash function. The sketch of a document is then its set of h minimum hash values and the similarity of two documents is estimated based on the overlap of their sketches [4]. In this study, we make use of hash functions in the form of: $h(x) = (Ax + B) \mod p$, where x is the shingle, p is a large prime, which we set to $2^{32} - 1$, and A and B are random integers in the range $[1, p]$.

To find near duplicates based on their sketches, each document is represented by pairs of the h minimum hash values - M_h - and the document ID in the form of $<M_h, doc\_id>$ and a list of all the pairs for all documents is compiled. This list is then used to build a second list of documents that have a M_h in common in the form of $<M_h, doc\_id_1, doc\_id_2>$. This second list can then be scanned and the number of M_h that each pair of documents $<doc\_id_1, doc\_id_2>$ have in common can be counted and then divided by h to calculate the resemblance of the two documents.

4. EVALUATION

To evaluate the algorithms, 100,000 documents were randomly sampled from the CiteSeer^x collection and those that contained a minimum of 15 tokens (after preprocessing) were retained, which was 95,558 documents. Each document was processed using standard information retrieval processing and the calculation of hashes and the extraction of shingles was based on the full text of the documents. No clustering took place when detecting near duplicates, since random pair sampling was used for evaluating precision and recall similar to the approach used in other studies [4, 6]. In deciding whether or not a pair of papers are near duplicates, the following should be true: the papers should have the same (or very similar) titles and authors; there should be significant overlap in the text (maximum of a paragraph different); and there should be significant overlap in the citations.

To evaluate precision for each treatment, which represents a variation in the parameters for duplicate detection, $n = 20$ pairs of documents that were identified as being near duplicates were randomly sampled. The n pairs were then manually checked in order to determine whether or not the documents were near duplicates and precision was calculated. Since no gold standard exists, it is impossible to accurately measure recall. Thus, recall is instead estimated by maintaining a list of all of the true positives identified during the precision calculation for each treatment, which we refer to as the *duplicate list*. The recall of each treatment is then estimated by comparing the documents returned by that treatment to the duplicate list. In total, the true duplicate list contained 360 unique duplicate pairs.

4.1 Detecting Duplicates

Simhash.

Experiments were conducted for different Hamming distance values of k, where $k = \{0, 1, 2, ..., 10\}$. Figure 1 (a) shows the the precision, recall and F-score. As can be seen from the figure, there is perfect precision when $k = 0$, which is to be expected since, in this case, each document has exactly the same hash. Thereafter, the precision decreases as k increases. When a Hamming distance of 5 is allowed, the

Table 1: Cosine similarity for different k

0	1	2	3	4	5	6	7	8	9	10
1	0.97	0.97	0.99	0.89	0.97	0.81	0.75	0.66	0.54	0.49

precision is approximately 0.65; however, precision decreases significantly beyond this point, ultimately ending up at less that 0.2 for $k > 7$. The recall also increases as k increases. This is to be expected since higher values of k allow for documents with larger Hamming distances between their hashes to be considered near duplicates. When $k > 3$, the recall exceeds 0.9. The F-score is highest at 0.91 and occurs when $k = 3$. At this point, the precision is 0.94 and the recall was 0.88. Interestingly, $k = 3$ was also the optimal value found for duplicate Web page detection [6].

Shingles.

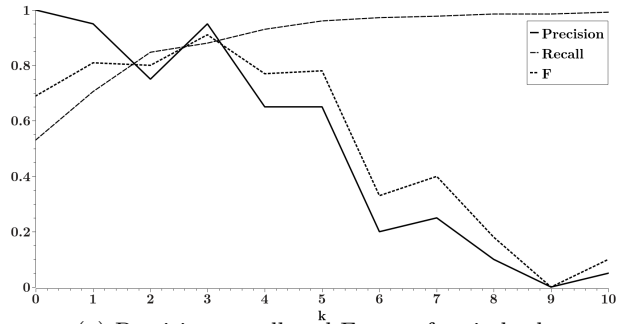
The number of hashes that were used to represent a sketch for a document was set to 84 as this has previously been used in other studies [1, 4]. Three different shingle sequence lengths $w = \{5, 8, 10\}$ were experimented with and the minimum required resemblance R for a pair of documents to be considered duplicates was also varied. Figures 1 (b) and (c) show the precision and recall. As can be seen from the figure, the length of the shingles and the minimum resemblance both have little effect on the precision, with each having a minimum precision of 0.95. As the resemblance requirement is relaxed, the recall increases. Furthermore, shorter shingle lengths result in higher recall, which is intuitive since shorter shingle lengths allow for more differences among the sequences of tokens that appear in the documents. The highest recall of 0.98 occurs when $w = 5$ and $R = 50\%$. The maximum F-score of 0.99 occurs when $w = 5$ and $R = 50\%$. At this point the precision is 1 and the recall is 0.98.

Thus, this experiment has shown that both algorithms perform well and, with the right parameters, can successfully be applied to detecting duplicate academic documents.

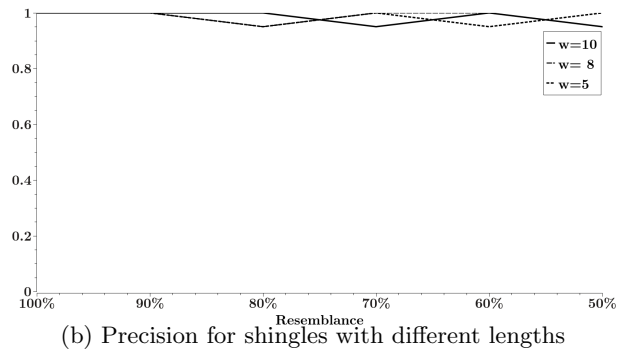
4.2 Analysis of Duplicates

We analyzed the cosine similarity between the $n = 20$ near duplicates pairs that were randomly sampled for each method and results are shown in Table 1 for different values of k for the simhash method. As can be seen from the table, as k increases the cosine similarity decreases. Interestingly, there is a large difference between the cosine similarity and precision in Figure 1 (a) for some k. For instance, for $k = 6$ the average cosine similarity is 0.81, but the precision is only 0.2. Since many of these papers are based on computer science, one possible reason for this could be due to significant overlap in common mathematical notation among papers, thus leading to similar hashes for different papers. Thus, the simhash method using single word tokens may not be appropriate for near duplicate detection for documents that make use of a large amount of standard notation. For the shingles method, regardless of the shingle size w or the resemblance, the minimum cosine similarity was 0.97 and was 1 in the majority of cases. This corresponds with precision of almost 1 achieved by the shingles method, regardless of the shingle length and resemblance.

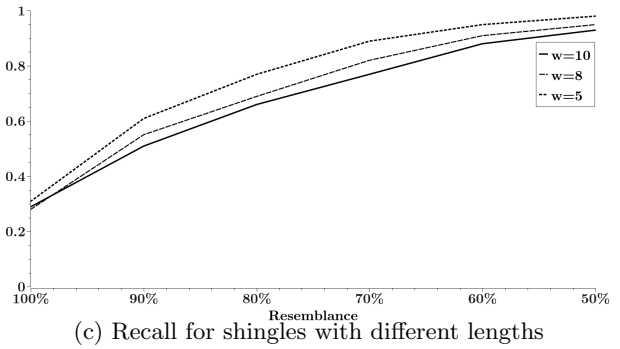
We also analyzed the types of duplicates returned by each method for the best performing parameters and labeled each true positive as being *exact*, a *preprint* (missing page numbers, copyright notice, different formatting, etc), or a *different version/draft* (minor differences in content, dates, and revision numbers). Table 2 summarizes the results of the types of near duplicates detected. As can be seen from the table, most of the duplicates returned by each method are in fact exact duplicates; however, some of them are also preprints and different versions/drafts of the same paper. The shingle-based method appears to return a more even distribution of different types of near duplicates, thereby suggesting that it is better than simhash at detecting different types of near duplicates.



(a) Precision, recall and F-score for simhash



(b) Precision for shingles with different lengths



(c) Recall for shingles with different lengths

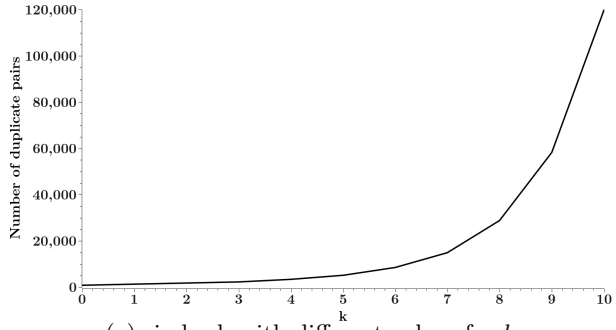
Figure 1: Performance of algorithms

4.3 Number of Duplicates Returned

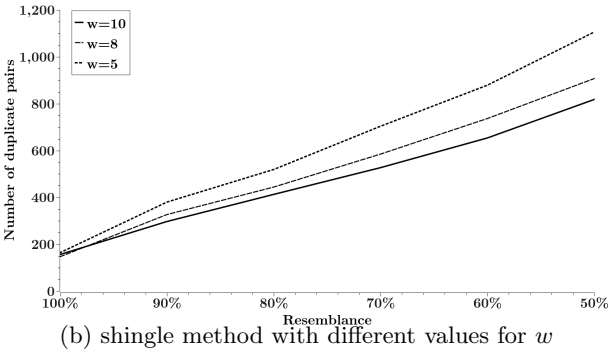
Figure 2 (a) shows the number of duplicates returned for simhash as k increases. When $k = 0$, 769 pairs of documents are returned. Thereafter, there is an exponential increase in the number of documents returned as k increases and it approaches 120 000 when $k = 10$. Based on the performance of the simhash algorithm, it is likely that the majority of documents returned when $k > 5$ are false positives.

Table 2: Types of near duplicates

Method	Exact	Preprint	Content/Version
Simhash	12	3	4
Shingles	9	5	6



(a) simhash with different values for k



(b) shingle method with different values for w

Figure 2: Number of duplicates returned

For shingles (Figure 2 (b)), lower values of w return more near duplicate pairs. This is in line with intuition, since lower values of w require that documents have shorter sequences of tokens in common and thus are more likely to match. Furthermore, Figure 2 (b) also shows that the number of duplicate pairs returned increases linearly as the similarity threshold is reduced. On average, reducing the required resemblance by 10% leads to 157.35 new document pairs being found. These results reveal something interesting about the shingle-based method, specifically, that near duplicate papers generally have high shingle similarities and reducing the similarity threshold does not lead to a large increase in the number of near duplicates returned. Given the performance of the shingles algorithm, it is likely that most of the near duplicates returned are true positives.

The results discussed above show that the two algorithms perform quite differently when the criteria for detecting duplicates are relaxed. For the simhash algorithm, increasing k leads to an exponential increase in the number of duplicates returned, whereas decreasing the similarity threshold for the shingle-based method leads to a linear increase in the number of duplicates returned. Thus, from the perspective of processing time, tuning these parameters is important so as to minimize the number of false comparisons made. As is shown in Figure 1, for the best parameter values we found, most of the comparisons made were between true positives.

5. CONCLUSIONS

We investigated the application of two state of the art duplicate detection methods to academic documents and identified parameters that could successfully be applied to achieve high precision and recall. We also analyzed the types of duplicates retrieved and, since the papers in the CiteSeer^x collection are collected through automatic crawling, this provides some evidence of the types of freely available academic documents on the Web. One question that arises is what should be done with these different versions of documents once near duplicates have been detected. For instance, they could be merged into a single record or all except a single copy could be deleted. In both cases, the question arises as to which version should be considered *authoritative*? We believe automatic document disposition could be used to address this problem. The goal would be to identify and rank duplicates based on pre-defined criteria and then take action in accordance with a policy that dictates how near duplicates should be treated. We also believe that it would be useful to investigate the use of different features for the detection of duplicates, for instance, different weights could be applied to different parts of a document, such as weighting the authors and titles of papers higher than the main text.

Acknowledgments

We gratefully acknowledge partial support by the National Science Foundation under Grant No. 1143921 and useful suggestions from Madian Khabsa and Sagnik R. Choudhury.

6. REFERENCES

[1] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic Clustering of the Web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, Sept. 1997.

[2] M. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 380–388, 2002.

[3] A. Chowdhury, O. Frieder, and D. Grossman. Collection Statistics for Fast Duplicate Document Detection. *ACM Transactions on Information Systems*, 20(2):171–191, 2002.

[4] M. Henzinger. Finding Near-Duplicate Web Pages. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 284–291, Aug. 2006.

[5] P. Lakkaraju, S. Gauch, and M. Speretta. Document Similarity Based on Concept Tree Distance. *Proceedings of the 19th ACM Conference on Hypertext and Hypermedia*, pages 127–132, 2008.

[6] G. Manku, A. Jain, and A. D. Sarma. Detecting Near-Duplicates for Web Crawling. *Proceedings of the 16th International Conference on World Wide Web*, pages 141–149, 2007.

[7] L. Padmasree, V. Ambati, J. Chandulal, and M. Rao. Signature Based Duplication Detection in Digital Libraries. *Signature*, 2006.

[8] I. Z. Yalniz, E. F. Can, and R. Manmatha. Partial Duplicate Detection for Large Book Collections. *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 469–474, 2011.