Near-duplicated Documents in CiteSeerX

Yi Zhang and **Jianguo Lu**
School of Computer Science, University of Windsor
401 Sunset Avenue, Windsor, Ontario N9B 3P4. Canada

Abstract

Academic literatures, especially those in the field of computer science, are often posted multiple times on the Web. Scholarly index engines, such as Google Scholar and CiteSeerX, crawl such documents from the open web as well as publishers. To improve the quality of the search result, there is a need to detect and coalesce duplicate or very similar (hereafter called near-duplicate) papers. Near-duplicate detection is computationally expensive. Pair-wise comparison of millions of papers is not feasible even for the most advanced machines. We combine SimHash and Jaccard similarity to discover near-duplicate documents in a CiteSeerX data set, which contains 2,118,122 full-text academic papers. We observe that 12% documents in CiteSeerX have near-duplicates with Jaccard similarity larger than 0.9. Then we study the near-duplicates and summarize six leading causes. We also compare these near-duplicates with those appeared only once on the web. We find that the citation count grows almost linearly with the number of duplications.

Introduction

Academic literatures, particularly the ones in computer science, often occur multiple times on the Web. Researchers may post drafts on their personal websites. Then, the same paper may occur in arXiv, or proceedings of conferences, etc. In addition, such documents may occur in course web pages. Each version may differ slightly, mostly in the publisher's information, or slight difference in content. Documents with slight difference are called near-duplicates. The threshold for the similarity depends on the application. This paper regards two documents are near-duplicates if their Jaccard similarity of their trigrams exceeds 0.9.

Detecting near-duplicates is an essential component of a search engine. Although the detection of near-duplicate of web pages has been studied extensively (Broder 1997; Broder et al. 1997), there are only a few works focus on the academic literatures (Williams and Giles 2013). The questions we want to answer are: 1) How many academic literatures on the web are near-duplicates? How similar are they? 2) What cause these near-duplicates? 3) What are the patterns of such multiple postings? 4) Whether multiple occurrences correlate to the citation number of the documents?

To answer these questions, we need to conduct an experiment on a large scale dataset. Our experiment is conducted on CiteSeerX datasets, which contain over two million full-text academic literatures. When detecting the near-duplicates, we want to tolerate slight difference between documents. Comparing the similarity in such a big dataset is computationally expensive. Thus, an efficient algorithm is required. In this paper, we first evaluate the state-of-art SimHash algorithm on the academic literatures. For web pages, it is reported to set Hamming distance $k = 3$ to achieve 75% accuracy. We find that the accuracy is higher in general for research papers. We also notice that the recall of SimHash can be as high as 99% when retrieving the near-duplicated literatures. By combining SimHash and Jaccard similarity, we successfully discovered 271,906 distinct near-duplicates with high accuracy, which contribute 12.84% of the CiteSeerX dataset. By studying these near-duplicates, we summarize 6 leading causes. And we also compare the categories and publishing years of the near-duplicated documents with the one appeared only once on the web.

We also observe that paper with more near-duplicates are cited more often. Moreover, it is interesting to see the citation counts grow almost linearly with duplicate occurrences.

Literature Review

Broder et. al. studied the near-duplicates on the AltaVista search engine by calculating the MinHash of the documents (Broder 1998). After that, numerous improvements have been proposed (Fetterly et al. 2003; Henzinger 2006; Hajishirzi, Yih, and Kolcz 2010).

SimHash is one of the most widely used near-duplicates detection algorithms, which is first introduced by Manku et. al. in 2007 (Manku, Jain, and Das Sarma 2007). After testing on three billion documents, the authors reported that the optimal Hamming distance of SimHash is three, which is the break-even point of precision and recall. Later on, Sood et. al. (Sood and Loguinov 2011) studied the recall of SimHash. They pointed out that SimHash can be faster and less space consumption by sacrificing a small percentage of recall.

Although near-duplicates detection is well studied, most of the existing works mainly focus on general documents, especially the web page crawled by search engine. Only a few works mention about the near-duplicates in academic literatures. In 2013, Williams et. al. used SimHash to re-

move duplicate documents in CiteSeerX(Williams and Giles 2013) and obtained F-score 0.91. Later on, they released a website called SimSeerX(Williams, Wu, and Giles 2014) for locating near-duplicate papers. However, they did not summarize the duplicate literatures.

Compared with the existing works, we give a better view of the near-duplicates in academic literatures. Our experiment discovers the near-duplicated documents with high accuracy. Most importantly, we analyse the features of the near-duplicates.

Near-Duplicates Detection

If two documents share terms in large quantities, we call them near-duplicates. A common technique for near-duplicate detection is to break documents into a sequence of consecutive tokens called n-grams (Broder et al. 1997). Then the similarity between the documents can be measured by Jaccard similarity. Given two documents, the Jaccard similarity between their shingles A and B is defined as

$$JS(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

When the Jaccard similarity of these two documents $JS(A,B)$ is larger than a threshold t, where t is a number that close to 1, we say these two documents are near-duplicates.

Generally speaking, Jaccard similarity is a good measure of the similarity of two sets (Henzinger 2006; Broder et al. 1997). Thus, in this paper, we treat Jaccard similarity as the ground true similarity between documents.

However, calculating Jaccard similarity directly is very costly, especially when we want to find near-duplicates in a large-scaled dataset. Sampling based techniques, such as MinHash(Broder 1997), has been proposed to reduce the computation time. SimHash, proposed by Manku et. al (Manku, Jain, and Das Sarma 2007), is one of the most widely applied near-duplicates detection algorithm. It maps the tokens of a document into a fixed bit fingerprint. Then the similarity can be estimated by the Hamming distance of the fingerprint. The algorithm can be described as follow:

- Split each document into n-grams (here we use trigrams);
- Initialize a K-dimensional zero vector V;
- Get a K-bit hash value for each trigram;
- For each hash value, if the i-th hash value is 1, then the i-th-bit of V increases by 1; if the i-th-bit hash value is 0, then the i-th bit of V decreases by 1;
- Then normalize V by marking the positive vectors as 1, and all the other vectors as 0. Thus, the fingerprint V can be stored as a K-bit integer.

After mapping each document into a K-bit fingerprint, where $K = 64$ in most applications, the similarity between two documents can be estimated by the Hamming distance of their SimHash fingerprints. Instead of computing all the possible pairs in the dataset, SimHash hash the fingerprints into certain tables to reduce the computation time. According to Pigeonhole principle(Herstein 2006), if n items are put into m containers, with $n > m$, then at least one container must contain more than one item. Thus, we can build an index for the SimHash values by splitting the K-bit fingerprint into $k+1$ sub-fingerprint. When we need to find the near-duplicates of a document, we can find all the candidates by fetching out all the related documents that have the same sub-fingerprints.

Parameters

SimHash requires a detection threshold – Hamming distance k between the similar documents. Two documents can be treated as near-duplicates if the Hamming distance between their fingerprints is less or equal to k. Most existing works use $k = 3$, which is first reported by Manku when detecting the near-duplicates among web pages (Manku, Jain, and Das Sarma 2007). However, scholarly documents are different from web pages. For example, web pages crawled from different websites may share the advertisement in the text; Academic documents do not overlap a lot due to the restriction of copyright; Academic documents are normally longer than web pages.

Next, we need to find out the performance of SimHash algorithm on scholarly literatures. In this paper, we collect a sample of 20,000 possible duplicated documents from CiteSeerX by matching their metadata. Then, we break the documents into trigrams and compute the pair-wised Hamming distances of the SimHash and the corresponding Jaccard similarities.

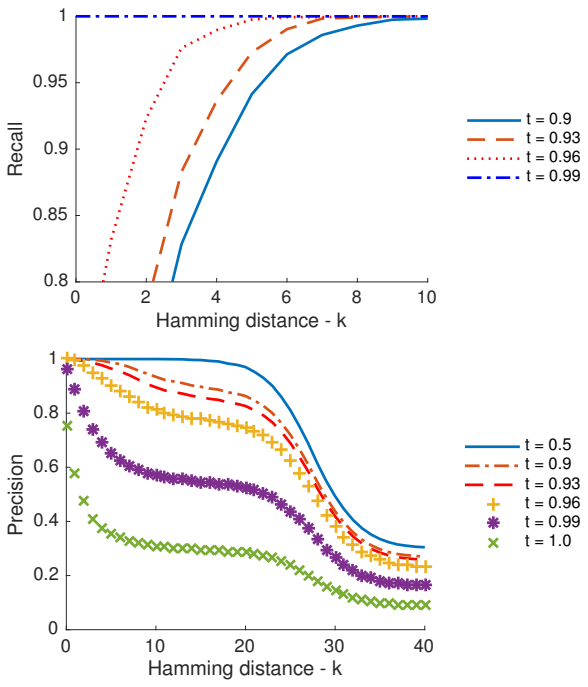


Figure 1: Precision and Recall with different threshold t

Fig. 1 shows Hamming distance threshold k against precision and recall. From the top panel, we observe that given a fixed detecting threshold t, the higher k is, the higher recall

we can achieve. For example, suppose we want to find near-duplicated documents that overlap 90%, when $k = 3$ the recall is about 75%, which means about 25% near-duplicates can not be found. In our work, we want to exam the features of near-duplicated academic literatures, thus, we need to get as many duplicated documents as we can. Therefore, a high recall is demand, which means we need to set k as high as possible.

However, a higher k may leads to two problems. The first is the precision. A higher k means less accuracy. Bottom panel of Fig. 1 shows that the precision decreases when a higher k is selected. The rate of deterioration accelerates when k becomes larger. The second, but larger problem is the time complexity. Fig. 2 shows the run time against k for different size of the datasets. Note that it is the execution time on a powerful server. Runtime increases exponentially with the growth of k. For a smaller data set that contains 20,000 documents, such growth of time is tolerable. For the dataset which contains 2 million documents, it needs around 6 hours to run if $k = 8$ and needs many days to finish if $k = 20$. Thus, in the later experiment, we set $k = 8$.

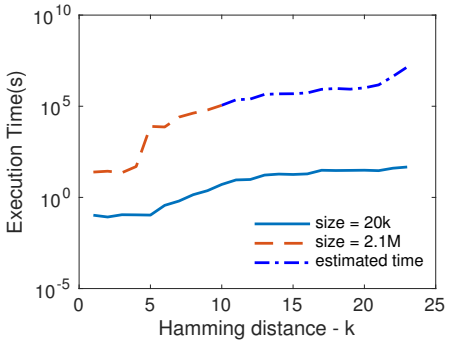


Figure 2: Runtime against Hamming distance k on different sizes of dataset

Next, we need to set a detecting threshold t for Jaccard similarity. Figure 3 shows precision and recall in different t with $k = 8$. As we can see, to guarantee the high recall of the near duplicates, particularly when documents do not overlap a lot, we need to set t as high as possible. But, a larger t could result in less number of near-duplicates. For example, when $t = 0.99$, we find 0.83 million pairs of near-duplicates. While $t = 0.90$, the number grows to 0.9 7 million. Our goal is to find as many near-duplicated documents as we can. In our work, we set $t = 0.9$ to balance the recall and the number of near-duplicates. With such setup we have a recall equals to 0.985 and accuracy of SimHash is 0.9581%. Meanwhile, to further boost the accuracy of the results, we calculate the Jaccard similarity of the document pairs captured by SimHash, and then save those have Jaccard similarity larger than detection threshold t.

Experiment and Results

Experiment Setup

CiteSeerX (Giles, Bollacker, and Lawrence 1998; Bollacker, Lawrence, and Giles 1998) is a scholarly index engine which

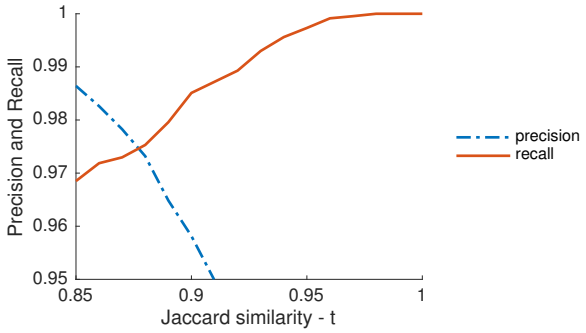


Figure 3: Relation between Jaccard similarity and precision / recall when $k = 8$

contains over 2 million full text academic literatures that are crawled from the open Web. It provides an OAI collection system that allows researchers download CiteSeerX data. The CiteSeerX data contains metadata, PDF files and corresponding text files that are extracted automatically using Prescript (Giles, Bollacker, and Lawrence 1998). In our experiment, we collect all the available text files (2,118,112 in total) from CiteSeerX OAI, then break these files into trigrams. Go-Language has been used to support parallelism computation. The program is executed on a PowerEdge R720 server that has 24 cores and 256GB memory.

Results

With detecting Hamming distance $k = 8$, the program takes 6 hours and 37 minutes to finish. In total, we discover 604,596 pairs of near-duplicates, which contains 364,930 distinct documents. The near-duplicates have been separated by their Hamming distance and Jaccard similarity in Tab. 1 and Tab. 2. From Tab. 1 we can see that most near-duplicated pairs have Hamming distance of 0. While it is interesting to see that they distribute evenly among other values. Meanwhile, most pairs concentrate in the range 0.9-1 as shows in Tab. 2, which means two documents are either very close or very different.

Hamming distance	pairs #	distinct documents #
0	397,112	79,461
1	19,767	35,666
2	23,165	42,147
3	24,441	43,903
4	25,437	45,998
5	26,483	47,368
6	27,237	48,576
7	29,298	51,362
8	31,656	54,790
sum	604,596	364,930

Table 1: Near-duplicates distribution by SimHash with $k = 8$

According to our previous evaluation, we can estimate the population of near-duplicates showed in Fig. 4. Because dif-

JS range	Pairs #	distinct documents #
0.9 - 1.0	524,888	270,906
0.8 - 0.9	51,161	61,847
0.7 - 0.8	19,826	22,491
0.6 - 0.7	6,164	6,518
0.5 - 0.6	1,491	1,576
0 - 0.5	1,066	1,592
sum	604,596	364,930

Table 2: Captured pairs and documents grouped by Jaccard similarity

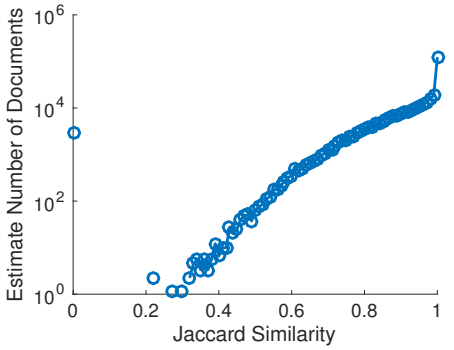


Figure 4: Estimated population of duplicated documents against Jaccard similarity

ferent Jaccard similarity ranges have different recall values, the estimations are augmented with different factors.

Discussion

Why do these near-duplicates appear multiple time on the Web? What cause the near-duplicates? To answer these questions, we randomly select the near-duplicates pairs and manually verified them. We summarize 6 leading causes of the near-duplicates as follow: 1) Same file publishes in different web pages. 2) Different stylesheet. Some literatures are written with LaTex. Different stylesheet, especially the format of citations and references, can cause slightly difference in the compiled PDF files. These near-duplicates often share large Jaccard similarities. 3) Different versions of documents. Some documents, books or reference manuals, have different published versions. 4) Error pages caused by the incorrect crawling process. CiteSeerX crawls the documents from the Internet. When crawling a web page, the server may not response correctly. Thus, an error page returns and is stored in the CiteSeerX database. These error pages from same website carry the same information, thus resulting in near-duplicates. 5) Extraction error. PDF is a common document format. CiteSeerX uses Prescript to extract raw text from PDF. However, not every PDF can be extracted correctly. Some PDF files have been extracted into short text files which contains only certain keywords (Introduction, abstract, References et.). 6) Class assignments also appeared in the CiteSeerX. These documents are not exactly academic literatures and should be removed from the collection.

We also notice that near-duplicates caused by incorrect crawling and extracting process. These documents sometime not share large similarities and are shorter than academic literatures. Meanwhile, they appear around one to two hundred times, which makes them easy to be detected and cleaned. It is possible to train a classifier to identify such near-duplicates, which we leave as the future work.

Next, we investigate the distribution of the duplicate occurrences showed in Fig. 5. The top panel in Fig. 5 shows the frequency of duplicate occurrences. From the plot, we can see that the distribution of duplicate occurrences follows a power-law. More than 100,000 documents only duplicate once (duplicate occurrence = 2) and more than 10,000 documents have duplicate occurrences = 3. Only a few documents have hundreds of near-duplicates.

The bottom panel is the duplicate occurrences against their ranks. It shows that the top one duplicated 807 times. The next repeats 195 times. We find that these files contain only one character. After that, there are a group of "nonsense" documents that are generated artificially to test Google Scholar's crawling and indexing strategy. They are repeated around one hundred times. These type of near-duplicates may be caused by incorrect crawling processing. Here, we removed these near-duplicates in the following observation.

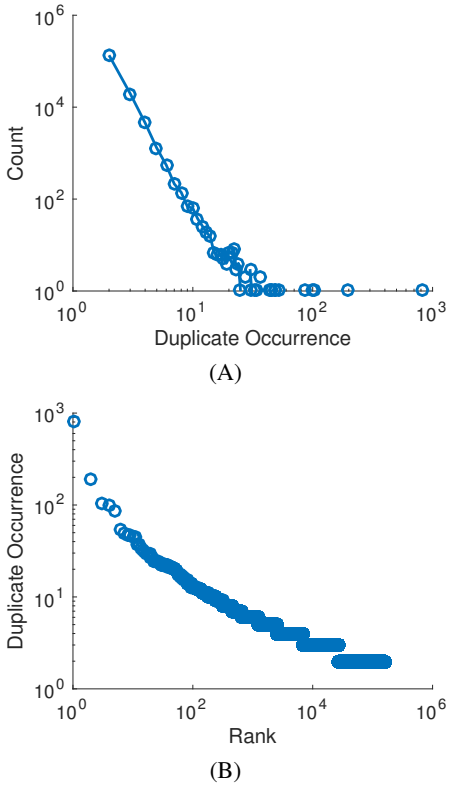


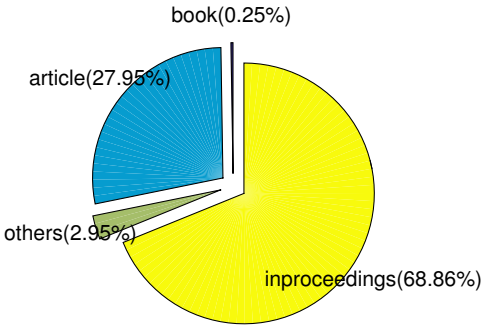
Figure 5: Distribution of near-duplicates.

Table 3 lists the top-10 duplicated documents in CiteSeerX. The most duplicated one is "Linearity and the pi-

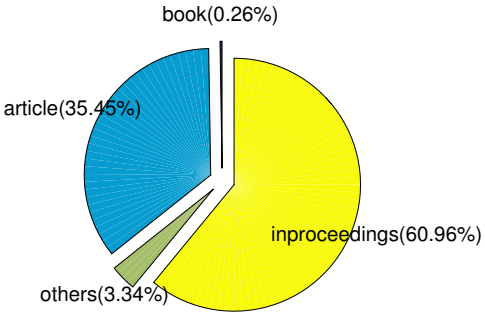
calculus" by Kobayashi, which repeats 49 times. Following is the "Serverless Network File Systems" and "Application Performance and Flexibility on Exokernel Systems". Interestingly, we notice that the duplicated documents not just include published books or papers, but also have some documents for the industry.

Then we list the difference of the document types between duplicated and non-duplicated literatures. Figure 6 shows the media types of the duplicated literatures. CiteSeerX split the document into 7 categories, and we keep the three major components and put the rest into "others". From the figure, we can see that "in-proceedings" contributes 68.86% in the duplicated documents and 60.96% in the non-duplicated ones.

The percentages of "books" of duplicated and non-duplicated documents are nearly the same(0.25%). While "article" contributes 27.95% in the duplicated documents and 35.45% in the non-duplicated ones. Category "others" in the top is 2.95%, which is slightly smaller than the one in the bottom.



(A) Duplicated documents



(B) Non-duplicated documents

Figure 6: Categories of duplicated and non-duplicated documents.

Next, we study the features of the near-duplicates. The top panel of Fig. 7 shows the distribution of publish years of duplicated and non-duplicated documents. It is interesting to see that they share the similar distribution. Documents that are published in earlier years have a higher chance to be seen, but they do not have many near-duplicates. The bottom panel shows the box plot of duplicate occurrence against publish years. We notice that older documents intend to have more near-duplicates.

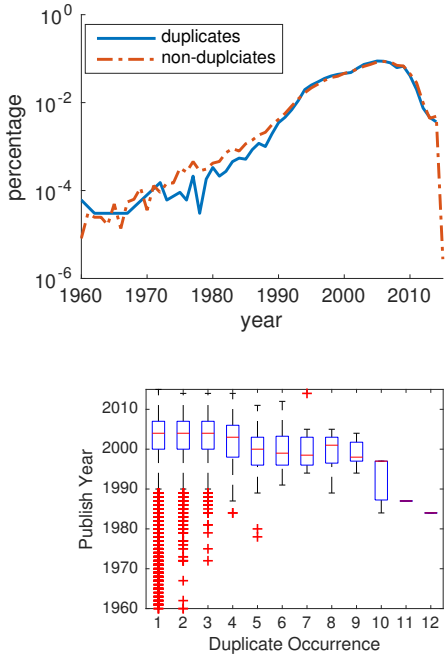


Figure 7: Publish years of documents.

Citation number is another important index to evaluate the quality of the literatures. We extract the citation count from the CiteSeerX metadata and show the relation between duplicate occurrence and citation count in Fig. 8. Panel A shows the average citation counts over duplicate occurrences. Note that there are a few outliers due to the sparse of the data when duplicate occurrences are large. According to the duplicate distribution in Figure 5(A), there are only about 20 data points for duplicate occurrences that are larger than 9. Thereby we ignore those sparse data and focus on the duplicate occurrences that are less than 9 in Panel B. From panel B we can see that the citation count grows linearly with the duplicate occurrence with Pearson Correlation 0.8894, particularly when the data points are abundance. Panel C is the box plot to show the dispersion of the data.

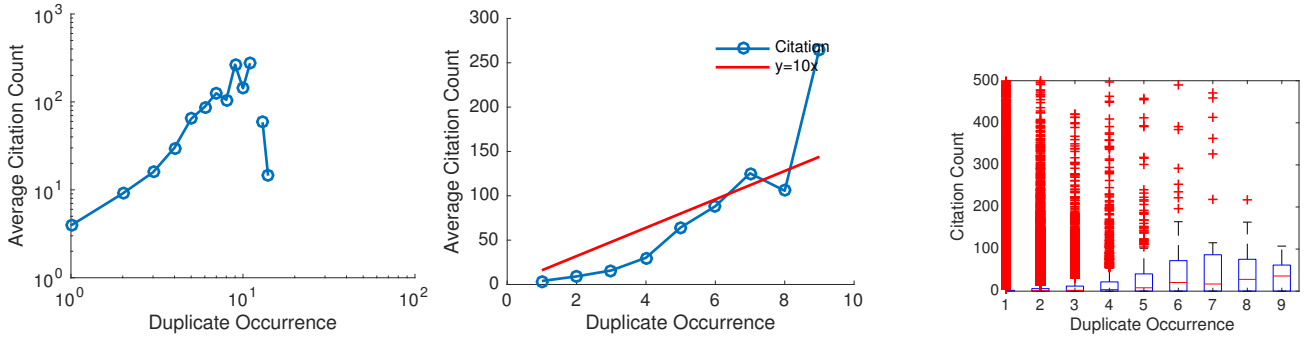
Conclusion

In this paper, we used SimHash and Jaccard similarity to detect near-duplicates in CiteSeerX. We found that SimHash needs to be set appropriate to balance the computational cost and accuracy. By combining SimHash and Jaccard similarity, we successfully retrieved most of the near-duplicates with Jaccard similarity larger than 0.9. We reported CiteSeerX has 12.79% documents duplicated more than once. This finding calls for further work to clean the CiteSeerX data for the construction scholarly search engines.

We also analyzed the near-duplicates, studied their features and made the observation of the relationship between duplicate occurrences with different features. The most interesting observation is that the citation count grows almost linearly with duplicate occurrence. This observation is important for both researchers and practitioners in search en-

Rank	Occurrence	Type	Document name
1	49	paper	Linearity and the pi-calculus
2	16	paper	Serverless Network File Systems
3	15	paper	Application Performance and Flexibility on Exokernel Systems
4	14	paper	A Fast File System for UNIX*
5	13	paper	A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols
6	13	article	Hints for Computer System Design
7	13	article	Security Architecture for the Internet Protocol
8	12	paper	Password Security: A Case History
9	12	paper	End-To-End Arguments in System Design
10	12	article	SWI-Prolog - Reference Manual

Table 3: Top-10 duplicated documents



(A) (B) (C)

Figure 8: Citation count against duplicate occurrence. (A) average citation count against duplicate occurrence; (B) Zoom-in of plot (A) ; (C) box plot.

gine industry. The gems of scientific works are often publicly available on the web in multiple locations. For researchers, it will be awarded if your publications are listed publicly on the Web. For the construction of academic search engines, we should crawl the open Web to find the gems in science.

Acknowledgement

This work is supported by NSERC Discovery program and Cross Border Institute. We would like to thank Zhou Tong for providing the citation count on the CiteSeerX dataset.

References

Bollacker, K. D.; Lawrence, S.; and Giles, C. L. 1998. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, 116–123. ACM.

Broder, A. Z.; Glassman, S. C.; Manasse, M. S.; and Zweig, G. 1997. Syntactic clustering of the web. 29(8):1157–1166.

Broder, A. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, 21–29.

Broder, A. Z. 1998. Filtering near-duplicate documents. In *Proc. FUN 98*.

Fetterly, D.; Manasse, M.; Najork, M.; and Wiener, J. 2003. A large-scale study of the evolution of web pages. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, 669–678. ACM.

Giles, C. L.; Bollacker, K. D.; and Lawrence, S. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, 89–98. ACM.

Hajishirzi, H.; Yih, W.-t.; and Kolcz, A. 2010. Adaptive near-duplicate detection via similarity learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, 419–426. ACM.

Henzinger, M. 2006. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, 284–291. ACM.

Herstein, I. N. 2006. *Topics in algebra*. John Wiley & Sons.

Manku, G. S.; Jain, A.; and Das Sarma, A. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, 141–150. ACM.

Sood, S., and Loguinov, D. 2011. Probabilistic near-duplicate detection using simhash. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, 1117–1126. ACM.

Williams, K., and Giles, C. L. 2013. Near duplicate detection in an academic digital library. In *Proceedings of the 2013 ACM Symposium on Document Engineering*, DocEng '13, 91–94. ACM.

Williams, K.; Wu, J.; and Giles, C. L. 2014. SimSeerX: A similar document search engine. In *Proceedings of the 2014 ACM Symposium on Document Engineering*, DocEng '14, 143–146. ACM.