Plagiarism Detection in arXiv

Daria Sorokina, Johannes Gehrke
Department of Computer Science
Cornell University, Ithaca, NY, USA
{daria, johannes}@cs.cornell.edu

Simeon Warner[1], Paul Ginsparg[1,2]
[1]Computing and Information Science
[2]Department of Physics
Cornell University, Ithaca, NY, USA
{simeon,ginsparg}@cs.cornell.edu

Abstract

We describe a large-scale application of methods for finding plagiarism and self-plagiarism in research document collections. The methods are applied to a collection of 284,834 documents collected by arXiv.org over a 14 year period, covering a few different research disciplines. The methodology efficiently detects a variety of problematic author behaviors, and heuristics are developed to reduce the number of false positives. The methods are also efficient enough to implement as a real-time submission screen for a collection many times larger.

1. Introduction

Improper reuse of text in academic research articles has a long, but largely undocumented, history. With the advent of widespread dissemination of electronic documents via the internet, the problem might be expected to worsen, due to the ease of obtaining and incorporating text written by others. On the other hand, those determined to appropriate text have long done so regardless of technology, whereas the greater ease of detection in the electronic realm may dissuade the less-determined. Full-text electronic research document corpuses have grown substantially over the past decade, and permit a systematic assessment of these issues. To our knowledge, there has been no systematic assessment even as to whether there is a background level of "borrowed" text snippets in a typical corpus which is ordinarily accepted by the community, and some threshold above which verbatim reuse would be regarded as inappropriate. In the following, we analyze these questions by making use of the arXiv: a document corpus that has nearly 100% coverage of certain research areas over an extended time period.

1.1. Background: the arXiv

The arXiv is an automated archive of physics, mathematics, computer science and quantitative biology articles [5, 15]. Since its creation in 1991, the arXiv has grown to over 375,000 articles (as of 7 July 2006), currently growing at a rate of more than 4000 new submissions each month.

The arXiv corpus is an excellent testbed for various analyses because the vast majority of articles are in formats from which the text content can be extracted. Over 95% of articles are supplied as TeX source (including LaTeX and other variants). These are automatically processed to produce versions for display, with PDF by far the preferred format at present. This process yields a set of PDF files amenable to text extraction. Experience has shown that taking the text from PDF is more practical than using the TeX source directly, because of the varied and complex macro substitutions possible within TeX; in general a complete TeX system is needed to interpret these.

There have been a small number of cases of plagiarism reported to arXiv administrators by readers, some quite egregious. The availability of efficient algorithms, as described here, means that it will be possible to automate the detection process, both to identify plagiarism in the existing corpus, and to provide real-time detection of plagiarized or duplicate articles at submission time.

1.2. Finding plagiarism and duplicates

Our first step will be to rank pairs of documents based on a notion of *similarity* between the two documents. By "duplicate", we refer to a document whose content is effectively an (improper) subset of another with overlapping authorship (see Section 3.2.1 for further discussion). Finding duplicates versus finding plagiarism requires different notions of similarity.

Consider, for example, two 15-page documents sharing one page of text. If the two have non-overlapping authorship and the articles do not reference one another, then sharing an entire page of text should be considered suspect, and these articles should be flagged with a high similarity score for the plagiarism task. If the two documents are written by the same author, however, this could simply be a case of different research on the same dataset, with the description of the data set copied from the author's own previous work, so this should not score high on either the plagiarism or duplicate-finding task. If the two documents share one or more authors, and one references the other, and if the overlap occurs in pages throughout one of the documents, then these two documents could be conference and journal versions of the same work, and should receive a high similarity score on the task of finding duplicates, but not on the plagiarism task.

1.3. Methodology

We now describe our approach for finding duplicates and plagiarism in the arXiv. The tasks differ, as noted above, but it is also clear that they share a common core capability: the detection of textual similarity in pairs of documents. We abstract this commonality by separating the two problems into a task-independent first stage, and a task-dependent second stage. First, we find all pairs of articles that have a significant amount of text in common. Then we select the subset of documents that satisfy further task-dependent conditions.

1.4. Outline

The remainder of this paper is structured as follows. We first outline in Section 2 our approach for finding pairs of documents that have similar passages of text. Then in Section 3 we show the results of an extensive experimental study that guided our parameter selection for the task-specific tuning of document similarity. Our results show that our algorithms are effective in finding both duplicate documents and plagiarism in the arXiv. We discuss related work in Section 4, and we conclude in Section 5.

We emphasize that the contribution of this article is not the methodology of detecting plagiarism or duplicates; in fact, there has been much previous work in this area, as we discuss in Section 4. To the best of our knowledge, however, this article describes the first large-scale application of methods for finding plagiarism in document collections, and it opens up interesting directions for future research.

2. Finding similar documents

We first discuss the problem of finding passages of text that are held in common by two or more documents. Our corpus is a collection of *documents*; each document has one or several *authors*. In our simple model, a document is made up of *sentences*, each consisting of a sequence of *words*.

First note that a set-oriented approach, in which each document is represented by a bag of words, is too coarse for this problem: two different articles on the same topic could have very similar bags of words regardless of whether they contain plagiarized text. We thus maintain the order of the words within a document, and use sentences as our unit of abstraction. Sentences provide a natural partitioning of text, though with the drawback that a single sentence is not limited in length. A plagiarizer could slightly modify a long sentence so that the new sentence is formally different, but would still be identified as plagiarism by a human reader. To address this problem, we introduce the notion of *similar* sentences as having overlapping consecutive parts of some fixed size. Our next two definitions capture this intuition.

Definition 2.1 *Two sentences are ν-**similar** if they contain the same sequence of ν consecutive words.*

Definition 2.2 *Two documents D and D' are **(ν, m)-similar** if there exist m sentences $s_1, \ldots, s_m$ in D, and m sentences $s'_1, \ldots, s'_m$ in D', such that s_i is ν-similar to at least one s'_j for $j \in \{1, \ldots, m\}$. Note that we do not require that these sentences are consecutive in either of the documents.*

Shared sentences between documents, however, do not always indicate plagiarism. Some common sentences are used by many authors. For example, the sentence "This paper is organized as follows." occurs in a large number of documents, and such common sentences should be excluded from our analysis. To take this into account, we refine our definitions as follows:

Definition 2.3 *A sequence of words is L-**common** if it is shared by at least L documents with non-overlapping authorship. We call a sequence of words L-*uncommon *if it is not L-common .*

Definition 2.4 *A sentence is (μ, L)-**common** if it contains an L-common sequence of μ words. We call a sentence (μ, L)-*uncommon *if it is not (μ, L)-common.*

Our definition of similar documents can now be refined as follows:

Definition 2.5 *Two documents are $(\mu, L; \nu, m)$-**similar** if both contain at least m different (μ, L)-uncommon sentences that are ν-similar to sentences in the other document.*

In the remainder of this section, we describe our approach for finding all document pairs that are similar by our definitions. Our approach is based on winnowing, a technique that has been previously used for plagiarism detection in programming assignments [11].

2.1. The winnowing algorithm

The winnowing algorithm is an instance of document fingerprinting: a document is summarized by a small set of character sequences called *fingerprints* which can be efficiently used to find copies of parts of a document in a large document collection. Document comparison is then reduced to finding exact matches in the sets of fingerprints. These sets of fingerprints should satisfy two main requirements: first, documents with overlap will be guaranteed to have intersecting sets of fingerprints; second, the sets of fingerprints should be small enough to permit scaling the task to large numbers of documents.

The winnowing algorithm [11] is one specific instance of a fingerprinting algorithm. For a given document, the algorithm first selects all contiguous subsequences of characters of length k, called k-grams. Note that a document of length n has $n - k + 1$ associated k-grams. Thus if two documents have a k-gram in common, then they have a common sequence of k characters. In practice, fingerprints are not the actual k-grams, but instead (unique) hashes of them.[1] The set of fingerprints associated to a document is reduced significantly by considering the document as a set of $n-w+1$ overlapping *windows* of length w, and retaining only enough hashes to ensure that each window contains at least one hash. If w is sufficiently large compared to k, the number of fingerprints associated to a document of length n is significantly smaller than $n-k+1$. It is straightforwardly shown [11] that even this reduced set of fingerprints can still find copies of sufficiently long passages of text, depending on the settings of parameters k and w: any match at least as long as the *guarantee threshold* $t = w + k - 1$ will be detected.

2.2. Text winnowing

Winnowing was developed for arbitrary digital documents. The winnowing algorithm can be adapted to our document collection by taking advantage of the background knowledge that the dataset consists of text documents that can be naturally segmented into words and sentences. This permits selecting a much smaller initial set of possible k-grams, and a restricted set of windows, which will both increase the speed of the algorithm and also permit working with a smaller but nonetheless effective set of fingerprints.

The differences between the text and original versions of winnowing are as follows:

- The minimum unit of text is a word instead of a character. This means that the fingerprint represents a sequence of k consecutive words rather than k consecutive characters, and that the size of the window is measured as w words, not characters.
- Each fingerprint is a subsequence of a sentence. Each window, from which a fingerprint is chosen, is a subsequence of a sentence as well. If a sentence is shorter than k words, it is ignored. If its length is greater than or equal to k, but less than w words, one k-gram is chosen from this sentence.

[1]Collisions in the hash function produce false positives.

The second refinement eliminates many windows and fingerprints that would have to be calculated in the original algorithm — those that cross sentence boundaries. It also permits a more reliable measure of text overlap than just the number of shared fingerprints. At one extreme, X shared fingerprints could correspond to a consecutive piece of only $k + X - 1$ words, because fingerprints can overlap. At the other extreme, when no fingerprints overlap, there may be X shared disjoint sequences of k tokens apiece. This is an inherent inconvenience of the original algorithm and is dealt with here by counting the size of overlap as the number of similar *sentences*, instead of the number of shared fingerprints.

We now describe how text winnowing can be used to find all pairs of documents similar by definitions 2.2 and 2.5. By setting $t = \nu$, we can guarantee that all ν-similar sentences will be detected. Then we calculate the number of sentences containing matching fingerprints and choose all documents that share at least m such sentences. The resulting set of pairs will contain all (ν, m)-similar documents. It is important to note [11] that when $k < t$ this set of pairs will be a superset of the target set. It will also contain some (but likely not all) pairs of (ξ, m)-similar documents for all $\xi : k \leq \xi < t$. The value of k determines the "noise-rejection" threshold, i.e., matches of fewer than k consecutive words will not be seen, so k is chosen as the minimum number of consecutive matching words that are likely to be of interest.

We can easily extend the approach to find all $(\mu, L; t, m)$-similar documents by taking $\mu = k$. After determining the number of files containing each k-gram in its fingerprint set, we identify a subset of all L-common k-grams, and therefore a subset of all (k, L)-common sentences. After eliminating them from the analysis, we are left with a superset of all (k, L)-uncommon sentences, forming a superset of all $(k, L; t, m)$-similar pairs of documents.

2.3. Data preprocessing

Most arXiv documents are stored either in PDF format (3.3%) or in TeX format (95.2%), which can be converted to PDF. In order to extract raw text uniformly, we use a PDF to text converter (PDFTOHTML [8]). We now describe the additional steps we took in some detail.

2.3.1. Parsing sentences

Additional processing is necessary before the PDF to text conversion output can be used as input for text winnowing. First we need to split the text into sentences. Periods and line breaks are possible markers of borders between sentences but there are some pitfalls: periods can appear in abbreviations and internet addresses ("Prof. Ginsparg","www.cs.cornell.edu"), and line breaks occur in the output of the PDF to text conversion not only at the end of a paragraph but in the end of every visual line of the original PDF document, i.e., most of the time in the middle of a sentence. We cannot ignore line breaks and consider periods alone, however, because headers and bullet items often do not end with periods.

We have chosen the following heuristics as rules for parsing:

1. A line break followed by a capital letter is considered an end of sentence.
2. A line break not followed by a capital letter is ignored.
3. A period at the end of common abbreviations such as "Prof.", "Dr.", "Fig." is ignored.
4. A period followed by a non-letter character (e.g., whitespace) is considered an end of sentence if the previous rule does not apply.
5. A period directly followed by a letter ("www.cs.cornell.edu" or "U.S.") is ignored.

This parsing is not perfect — it sometimes breaks sentences. The most problematic case we encounter is when a proper noun appears in the middle of a sentence and happens to appear in the beginning of a line in the PDF document. Then the first rule applies and the sentence is split. An uncategorized abbreviation in the third rule can also split the sentence. The latter case is not very dangerous here since identical sentences in different documents will be broken in the same place. The former case, however, can have a negative effect, because line breaks in the same sentence can happen in different places in different PDF files, so the same text might be parsed differently, resulting in a missed match.

We also made some effort to recover words split into several parts after PDF to text conversion. First, a hyphen followed by a line break is recognized as a division of a word, so we re-assembled the two word fragments into a single word. Second, we apply a set of rules specific to the output of the PDF to text converter to recover words with non-English characters such as ä, æ, ø, which appear corrupted but are systematically recoverable.

2.3.2. Text cleaning

After sentence parsing, we further simplify the text in each sentence as follows:

- Sequences of whitespace characters are collapsed to a single space
- All characters except letters and spaces are removed
- Words consisting of a single letter are removed
- All letters are converted to lowercase

This further text cleaning improves our ability to detect cases of interest without introducing false matches. During this process most formulae and variables are removed from the text. This is done on purpose: identical formulae by themselves do not mean that the text was copied.

2.3.3. Separating the bibliography

The bibliography can constitute a large fraction of a document. Text overlaps from this section are not of interest for similarity detection since we expect citations to be written similarly or identically in differently authored documents. The bibliography may contain additional useful information, however, such as the name of an author from whose article text has been copied. During the preprocessing phase, we attempt to find the beginning of the bibliography in the text representation of each document by searching for the line "References" or "Bibliography". Since documents containing a table of contents may have an additional single "References" line, the last occurrence of such a heading is used to split off the bibliography.

Empirical evidence shows that we succeeded in extracting the bibliography from about 75% of the documents. The most common reason for the failure of this simple strategy is the absence of "References", "Bibliography", or any other characteristic header.

2.3.4. Extracting author names

A crucial preprocessing task is correct extraction of authors' names. When comparing two documents, it is necessary to know if they are written by the same author in order to determine whether any overlapping text is a question of plagiarism or instead self-plagiarism/duplicate article. Fortunately, all arXiv documents have metadata containing tagged names of authors, text of abstracts, and other useful information.

Unfortunately, even after extracting names from metadata, in many cases there is still uncertainty about whether or not two articles are written by the same author. One problem is that many authors, especially from countries whose languages use other than the Latin alphabet (or use the Latin alphabet with additional letters), transliterate their names differently in different articles. We identify a set of different transliterations, taken from existing arXiv documents (Table 1).

The second major problem for name extraction involves large collaborations. Many arXiv documents, especially in

Russian	German	Armenian
ou = u	ae = a	ian = yan
ss = s	oe = o	
tch = ch	ue = u	
’e = yo		
zh = j		
ine = in		
ski = sky = skii = skij = skiy		

Table 1. Equivalent transliterations in names

astrophysics and high-energy physics, represent the work of large groups of people. In many cases, not all members of a collaboration are listed as authors. Sometimes they are all listed as authors in the text of the article, but in the metadata only one or two names appear, together with a parenthetical note “for XXX collaboration”. Sometimes there is only a single author followed by “et al” in the metadata, with no explicit collaboration name given. These cases are problematic, because it turns out that there is a large amount of reuse of text in reports by collaborations, typically by members of the collaboration writing single-author papers for independent conferences. Since it is important to distinguish texts created from within the same collaboration from actual cases of plagiarism, we applied the following heuristics. First, if the word “collaboration” is present in the “Authors” field of the metadata, then we combine the collaboration name into the list of extracted author names. Second, we semi-manually created a list of the most common collaborations, and scan the entire metadata, including abstract, for any such names. If a known collaboration name is present anywhere in the metadata, then this collaboration name is included in the list of authors as well. Standard text will not ordinarily be mistaken for the name of a collaboration because collaboration names are written in all uppercase, which we distinguish in this analysis. For example, the word “chess” will not be mistaken for the “CHESS Collaboration”.

2.4. Implementation

It is not efficient to store the full sequences of characters corresponding to our text fingerprints in memory. We instead convert them to 64 bit integers. As in the work of Broder [2], we use a hash function based on Rabin fingerprints [9]. A 64 bit hash was sufficiently large to avoid hash collisions (different fingerprints assigned the same integer).

The preprocessed text documents with one sentence per line are read line-by-line, and the fingerprints are stored as 64 bit integers in a hash table. The key for each entry consists of the first 32 bits of the fingerprint (multiple entries with the same key are allowed) and the value of the entry is a structure that contains the rest of the 64 bit fingerprint plus the id number of the document containing it.

Once the entire hash table is assembled, each entry can be analyzed to distinguish those fingerprints that are shared by several documents from those that are unique. At this stage, we also eliminate very common fingerprints, those shared by many documents and likely to be common phrases rather than explicitly borrowed text.

From the resulting table, we create a set of pairs of documents that share uncommon fingerprints for further analysis. It is important that this set of pairs is far smaller than the set of all possible pairs of documents, since all pairwise comparisons would be infeasible for the large arXiv dataset. We decrease the number of pairs further by selecting only those pairs that overlap by at least m uncommon k-similar sentences. In the final stage, we analyze each pair from the list and identify those most relevant for our purposes, by methods detailed in Section 3.2.

2.5. Parameter setting

The experiments for choosing reasonable parameters k and t of the winnowing algorithm were run on a subset of the data, selecting only 7200 articles from the “physics” subject area of arXiv. Parameters m and L were set to 4 based on intuition and computational restrictions. Later we did some experiments that supported our choice of L (details in Section 2.5.3).

The general framework of these experiments was the following: We set $m = 4$, looking for documents that share at least four sentences containing uncommon k-grams, and ran our algorithm many times with different values of k and t. We then manually assessed the differences between the sets of results to determine whether those pairs should be included or not, i.e., whether they constituted actual plagiarism or were false positives. These observations were used to select parameter values that screened as many false positives as possible, without losing too many true positives. Manual assessment of these differences was very time consuming so we used only a small subset of documents. The values chosen appear to be adequate for present purposes, but a more comprehensive assessment would be required to determine optimal values. We defer this investigation to future work.

2.5.1. Choosing the noise reduction threshold k

The first set of experiments was run with $k = t$ for $k = 4, 5, 6, 7, 8, 9$, in order to choose the most appropriate k. With these settings, the fingerprint sets contain all possible k-grams, so the results of each successive experiment are always a subset of the previous one. After each experiment, the reported cases were ranked by the size of overlap. Then the five top cases that did not survive increasing k were examined. The results are shown in Table 2: the

T values	deleted FP	deleted undefined	deleted TP
$4 \to 5$	5	0	0
$5 \to 6$	5	0	0
$6 \to 7$	4	1	0
$7 \to 8$	1	1	3
$8 \to 9$	0	0	5

Table 2. Influence of K - size of K-gram

T values	deleted FP	deleted undefined	deleted TP
$7 \to 8$	5	0	0
$8 \to 9$	3	1	1
$9 \to 10$	2	0	1
$10 \to 11$	3	0	0
$11 \to 12$	5	0	0
$12 \to 13$	0	1	4
$13 \to 14$	0	0	0

Table 3. Influence of T - window size

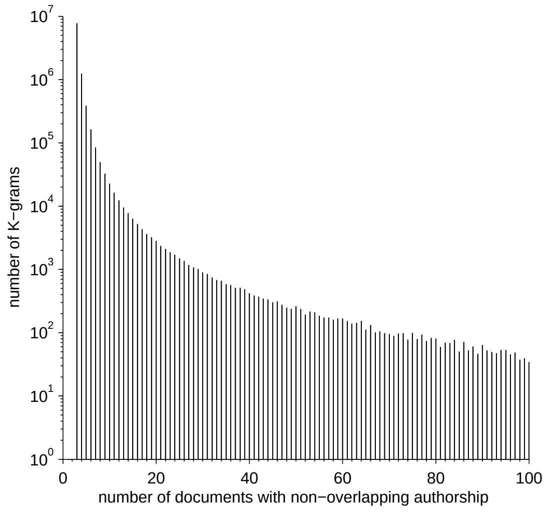

Figure 1. Distribution of K-grams over the number of authors sharing them.

first column shows the change in k, and successive columns show the number of false positives, hard to classify cases, and true positives from the top five deleted cases.

From these results, we infer that $k = 7$ is the most appropriate value for k for our purposes. Larger values lose true positives, and smaller values introduce false positives without adding true positives. We still have some false positives for $k = 7$, but we would prefer to retain all true positives at the expense of including some false positives.

2.5.2. Choosing the guarantee threshold t

After setting $k = 7$, we ran a new set of experiments to determine the optimal value of t. The quality of detection turns out to improve with increasing t. Formally, it is possible that each case present in the results with $t = k$ will be present in the results with the same k and larger t. That probability increases according to the overall similarity between the two articles. Increasing k, we found that the least similar cases were removed, and up to some point those turned out to be false positives. Similarly, we increased t for fixed k and looked at the five top-ranked results that disappear after this transition. If there were fewer than five such results, we looked at the whole set available. Results are shown in Table 3.
From these results, we inferred that $t = 12$ is a sensible choice, corresponding to a window size $w = t - k + 1 = 6$.

2.5.3. Common k-grams. Parameter L

Figure 1 shows on a log scale the distribution of k-grams, for $k = 7$ words, against the number of articles, with non-overlapping authorship, sharing them. The horizontal axis is the number of articles with non-overlapping authorship l (from 2 to 100), and the vertical axis is the number of k-grams shared by exactly l articles with non-overlapping authorship.
The first bar on this histogram shows that there are more than a million k-grams shared only by two articles with non-overlapping authorship — these are the units of overlap of primary interest for us. We include in the analysis the k-grams represented by the second bar — those shared by three articles with non-overlapping authorship, so that we don't miss the cases in which two different plagiarists copy from the same source. When we include k-grams from the third bar (shared by four different articles with non-overlapping authorship), the results included sets of documents overlapping only by a section of boilerplate text (such as standard institutional copyright paragraphs). Based on these results, we chose to set the parameter $L = 4$ in our algorithm, to identify 7-grams used by four or more articles with non-overlapping authorship as common.

Given all of the above, together with the discussion at the end of Section 2.2, our algorithm ensures that we discover all (7,4;12,4)-similar pairs of documents.

3. Experiments

In our experiments, we used a collection of arXiv articles from 1991 through early 2005. This dataset included 287,857 articles in PDF and TeX format. Out of those, 3023 were unusable due to conversion problems. The experiments were run on a single CPU (64 bit, Itanium 2,

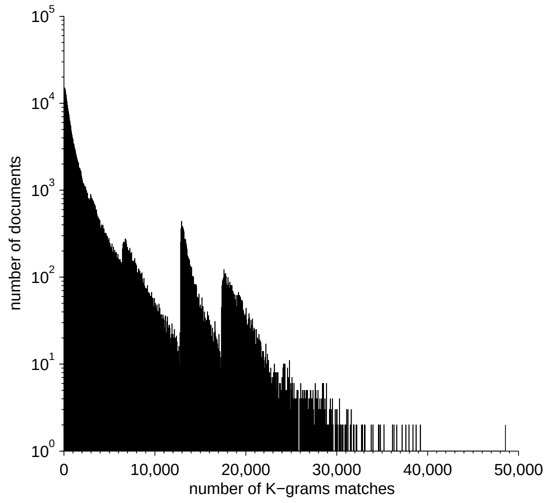

Figure 2. Distribution of 7-gram matches with multiple counting. All 7-grams.

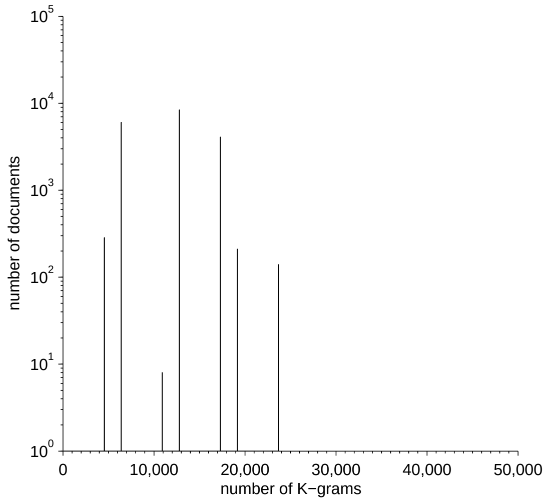

Figure 3. Distribution of 7-gram matches with multiple counting. Three most common 7-grams.

1.3 GHz) with 64 GB RAM. After preprocessing, the final run of the text winnowing algorithm and output of results took 20 hours. In the first step, 204,828,778 7-grams were entered into the multiple entry hash table, on average 721 7-grams per document. A set of 440,224 pairs of documents, each sharing at least 4 potentially uncommon 7-grams, was identified, and then reduced to 330,306 pairs sharing at least 4 potentially interesting similar sentences. This is our superset of all pairs of (7,4;12,4)-similar documents. 312,685 of them are pairs of documents created by the same author (duplicate candidates), and 17,621 are by different authors (plagiarism candidates).

3.1. Distribution of k-gram matches

Here we examine the distributions of k-gram matches in the data, for the chosen $k = 7$. The horizontal axes in the plots of this section show the number of k-gram matches (shared k-grams). The vertical axes, plotted on a log scale, show the total number of documents sharing the given number of k-grams. Each vertical bar corresponds to the total for a bin of width 50 (plus 1, so that the log can always be taken). Note that the number of shared k-grams includes repeat counting of any k-gram that co-occurs in many documents. For example, a particular document will be counted in the 1001–1050 bin if it shares 1025 k-grams with one other document, or 1 k-gram with 1025 other documents.

Figure 2 shows the shared k-grams for the whole corpus of 284,834 documents, with all winnowed k-grams included ($\sim 2.9 \times 10^8$ total k-gram matches for an average of $\sim$1000 per document). At the left edge, we see that 15,313 documents share 1–50 k-grams, and at the far right we see that just one document shares 48,510 k-grams with other documents.

While the overall trend is a decrease in the number of documents with increasing number of shared k-grams, the figure also shows additional structure of sharply rising and slowly declining peaks. Each such peak corresponds to one of the most common k-grams or a combination thereof. For example, the most common 7-gram “this work was supported in part by” is shared by 12966 documents. This means that each of these documents shares at least this 7-gram with 12965 other documents, and thus the total number of 7-gram matches with the rest of the corpus is at least 12965. The peak, however, is not a single 12965 document high spike because many of these documents also share other 7-grams with other documents, smearing the peak rightwards to higher numbers of 7-gram matches.

This explanation applies only if the numbers of documents sharing the most common k-grams are large compared to the average number of k-gram matches, otherwise the distribution would be smooth. The second most common 7-gram “can be expressed in terms of the” is shared by 6541 documents, the third most common 7-gram “work was supported in part by the” is shared by 4612 documents, and all other 7-grams are shared by less than 3000 documents. These most common 7-grams are shared by significantly more documents than the average of $\sim$1000 matches per document, and thus show as clear peaks. Of the three most prominent peaks, the first peak around 6500 corresponds to the second most common 7-gram, the second

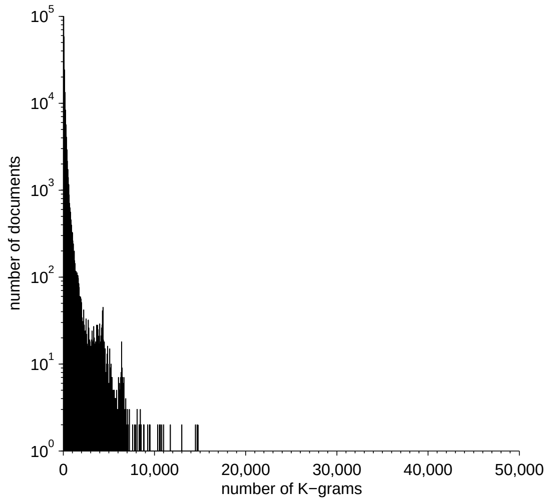

Figure 4. Distribution of 7-gram matches with multiple counting. Interesting 7-grams.

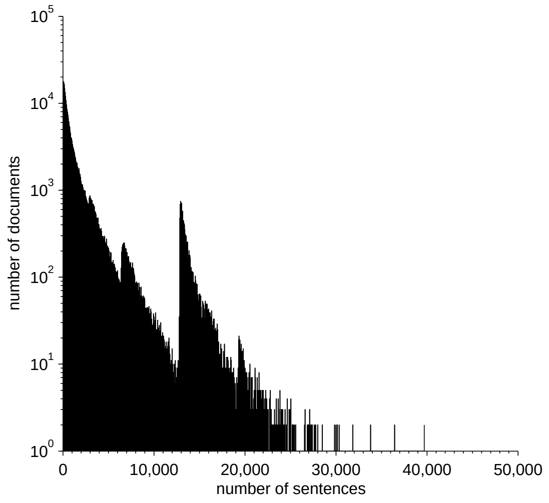

Figure 5. Distribution of sentences with multiple counting.

peak around 12900 corresponds to the most common 7-gram, and the third peak around 17400 corresponds to the documents sharing both the most common and the third most common 7-gram, as expected because those two 7-grams overlap.

Figure 3 highlights the contribution from common k-grams by showing only the contributions from the three most common 7-grams. The horizontal positions of the bars correspond to the most visible peaks of the previous figure. The positions of the peaks, from left to right, correspond to 3^{rd} most common k-gram, 2^{nd}, $2^{\mathrm{nd}} + 3^{\mathrm{rd}}$, 1^{st}, $1^{\mathrm{st}} + 3^{\mathrm{rd}}$, and $1^{\mathrm{st}} + 2^{\mathrm{nd}} + 3^{\mathrm{rd}}$.

Common 7-grams are not only responsible for the peaks, but in fact constitute much of histogram. Figure 4 shows uncommon 7-grams only, i.e., those appearing in at most three documents. This graph shows a fairly smooth drop off in number of documents for increasing numbers of 7-gram matches. There is still some structure for large numbers of 7-gram matches, corresponding to matches in significant portions of documents.

Finally, figure 5 depicts counts of sentences instead of 7-grams, and shows a smaller number of peaks than figure 2. This is because of the high correlation of the first and third most common 7-grams responsible for the peaks in figure 2. When the two 7-grams “this work was supported in part by” and “work was supported in part by the” occur in one document, they typically occur in the same sentence, so the sentence counts in figure 5 show no peak for the combination of these two 7-grams.

3.2. Extracting interesting documents

After creating the superset of all (12,4;7,4)-similar document pairs, we then extracted the subsets relevant to different tasks. As mentioned earlier, we will focus on the two problems of duplicate and plagiarism detection. We apply different rules to extract the corresponding subsets.

3.2.1. Duplicate detection

Duplicate here refers to an article essentially the same as another by the same author, or a subset of another by the same author. It can be a conference paper later extended to a journal article, or vice versa, or an article whose content is included in a PhD thesis, and so on. Detection of duplicates is helpful for improving search results: if there are many versions of the same article in the repository, and this article is relevant to the search query, then it is possible to prune (or group) duplicates in the results for the benefit of the reader.

The primary rule for a pair containing a duplicate article is clear: the two articles should have at least one author in common. This rule is applied after preprocessing is completed, so the common author can be either a collaboration or a specific individual.

Next we need to specify a criterion for determining when a pair of highly similar documents with an author in common is considered to contain a duplicate. The document must share similar sentences in every part of its text in order to be considered a duplicate. On the other hand, we are not interested in pairs in which both documents possess large consecutive parts of unique (not shared) text, even if they share chunks of some other text. We formalize this notion of duplicate in the following way:

Definition 3.1 *In a pair of similar documents,* ***the largest copy-free part*** *of one document is the longest sequence of consecutive sentences that includes no sentences similar to any sentence in the other document. The* ***originality*** *of the document is the ratio of the length of its largest copy-free part to the length of the whole document, where length is measured in number of sentences. The document is considered a* ***duplicate*** *if its originality is less than some α.*

In our experiments we have set α to 0.2.

3.2.2. Plagiarism detection

In plagiarism detection, unlike the case of duplicate detection, the size of the unique part of the document does not matter. The number of similar sentences is itself a good measure of plagiarism similarity. This task, however, turned out to be nontrivial in a different regard: there are many reasons why shared text in two documents by different authors might not turn out to be plagiarism. A number of additional heuristics had to be applied in order to extract a set of pairs containing only a few false positives (i.e., non-plagiarism cases). Indications of possible false positives are:

1. An author of one document is a neighbor on the coauthor graph to an author of the other document. People from the same research group can both reuse text from an earlier article, on which they were co-authors.
2. An author of one article appears in the text of references of the other. This is usually an indication of "mild plagiarism" — people who maliciously claim someone's work as their own normally do not cite their source.
3. An author of one article appears in the text of the other article. This can result from any of:
   - not all authors are properly indicated in the metadata
   - there is a direct citation from the other article
   - the first document is a workshop proceedings and the second is from the workshop
   - one of the actual authors is mentioned only in the acknowledgments
4. One or both articles are produced by a collaboration. Although significant effort was made to solve the collaboration problem during the preprocessing step, collaborations remain a probable source of false positives.
5. Both previous conditions (3,4) hold: this is a very strong evidence of a false positive, and occurs for example when all actual authors are listed in the full text of the article but not in the metadata.

heuristic	affected cases	impact
1. coauthor	8934	50.7%
2. referenced	6590	37.4%
3. mentioned	13148	74.6%
4. collaboration	2973	16.9%
5. ment. & coll.	2116	12.0%

Table 4. Heuristics applied to similar pairs of documents written by different authors

These heuristics can be too strong, however, and could result in elimination of true positives. The pairs flagged by either heuristic 1 or 5 are discarded as false positives, but those flagged only by any combination of 2,3,4 are retained in a secondary set of results. The primary list contains only those pairs with strong evidence of plagiarism, and the secondary contains mainly false positives, but some cases of interest.

The impact of each heuristic is shown in Table 4. The second column lists the number of affected cases and the third shows the percentage of the affected cases in our set of pairs of overlapping documents with non-overlapping authorship (17621 pairs).

3.3. Results

3.3.1. Common k-grams

We found the document corpus to contain 429,258 common 7-grams: consecutive sequences of 7 words shared by at least 4 documents written by different authors. Table 5 shows the ten most frequently occurring k-grams, those that appeared in the largest numbers of documents. The numbers in this table count the appearances of these k-grams as (winnowed) fingerprints, and therefore can underestimate the true total numbers of appearances. These common 7-grams are instantly recognizable to people familiar with research literature in the subject area, and typically fall into classes such as: describing the structure of the article; describing equations, figures, tables; parts of common descriptive research phrases; acknowledgment text; or institutional affiliations.

3.3.2. Duplicate detection results

We extracted 38,580 pairs of documents, at least one of which had level of originality with respect to the other at most $\alpha = 0.2$. After eliminating documents reported several times — this happens when there are many copies of the same work — 30,316 unique duplicates were left. This suggests that documents containing significant self-plagiarism and subsumed text constitute as much as 10% of the arXiv

7-gram	Documents	Authors	All occurrences
this work was supported in part by	12966	2085	13161
can be expressed in terms of the	6541	2460	7379
work was supported in part by the	4612	1015	4760
first term on the right hand side	3337	1524	4000
it is easy to see that is	2974	1452	3605
operated by the association of universities for	2900	396	3372
department of physics and astronomy university of	2880	539	3018
the paper is organized as follows in	2764	1202	2766
there is one to one correspondence between	2418	1316	2967
term on the right hand side of	2404	1194	2846

Table 5. Most popular 7-grams

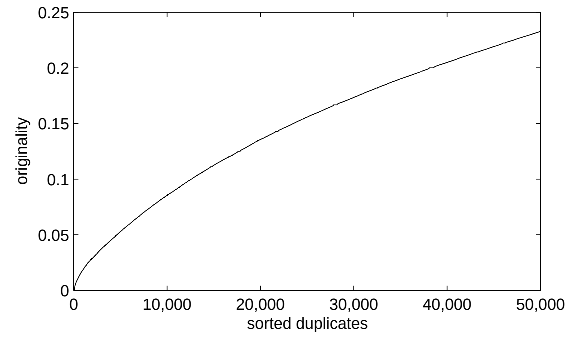

Figure 6. Reported duplicates sorted by level of originality.

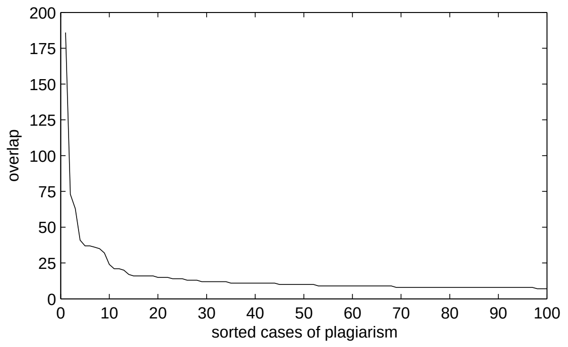

Figure 7. First 100 cases of plagiarism sorted by size of overlap.

corpus. Most of these redundant texts are apparently intentional, but this analysis also uncovered 64 identical duplicates. Many of those appear to have resulted from some form of submitter error, for example someone trying to submit two different articles, with different metadata, but accidentally submitting the same full text.

Figure 6 shows how the level of originality grows on the first 50000 pairs, ordered by the level of originality of the lower originality document in each pair. The smoothly increasing function suggests that there is no way to choose a natural threshold, separating duplicates from arguably different documents. The $\alpha = 0.2$ threshold was chosen here based on nominal intuition, but does not appear to result in false positives. There may nonetheless exist articles which would be considered duplicates, despite the higher level of originality, which we fail to detect.

3.3.3. Plagiarism detection results

Applying the heuristics of Section 3.2 to screen for false positives or "mild" plagiarism left 677 pairs of documents with at least four sentences sharing uncommon 7-grams. Detailed manual analysis of the first 20 pairs from this list revealed that 16 of them constitute clear cases of plagiarism. The 4 false positives resulted from i) two articles by the same author with two non-standard transliterations of his name, ii) two proceedings of different workshops sharing an article written by the same author, iii) two articles quoting the same text from Einstein, and iv) two articles with significant overlap in references which had not been automatically separable from the actual document texts.

Of the first 16 true positives, at least 3 appeared to be serious plagiarism, in which an article was essentially a copy of another written by others, albeit with many small text modifications. Many of the others were cases of articles and theses with an introductory or related work section appropriated from other sources, without even referencing them.

Figure 7 shows the size of overlap (number of sentences sharing uncommon 7-grams) in the top 100 of the 677 flagged cases. Although there are relatively few cases with large chunks of text plagiarized, the number of cases becomes large as the size of the overlap decreases. The last 535 of the 677 cases came from 79 cases sharing 6 similar sentences, 138 sharing 5, and 318 sharing 4.

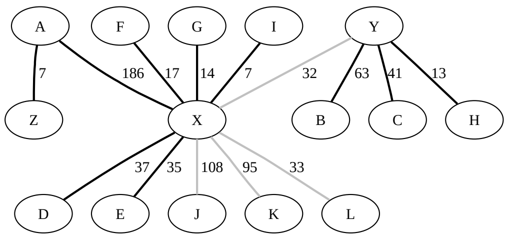

Figure 8. Subgraph of the overlaps graph. Two major plagiarists.

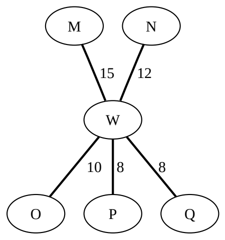

Figure 9. Subgraph of the overlaps graph. A new case of real plagiarism.

The 677 flagged pairs contain 1086 unique articles. Some of those appearing in multiple pairs are cases of a document plagiarizing from more than one source, and some from a source plagiarized multiple times: plagiarists often copy from many sources, and some sources, such as lecture notes, are attractive to many plagiarists.

There were an additional 7371 cases in the secondary list, removed from the primary list by the heuristics of Section 3.2. These are mainly false positives, but some examples of unethical behavior also appear. In the top 20 pairs, three were cases in which tens of pages were copied into a thesis without any modifications, although in these cases the sources were at least acknowledged in the text. (Proper usage, however, would require the entirety of the offending text to be placed in direct quotes.)

The secondary list contains 10,072 unique documents and the two lists together contain 10,763 unique documents.

3.3.4. Visualization

To identify interesting cases of plagiarism, it is useful to employ a graphical representation of the lists of document pairs. Many plagiarized articles contain text from different sources, and therefore appear in many entries of these lists. If the list is large, it is difficult to estimate the extent of plagiarism in a single article. To overcome this problem we have visualized the results as a graph of overlaps: each node is a document, edges connect overlapping documents, labels on edges show the number of shared similar sentences. Black edges correspond to the primary list of results (most probably plagiarism), grey edges correspond to the overlaps reported in the secondary list (“mild” plagiarism, if any at all). We have masked the ids of actual arXiv documents with single-letter labels: A, B, C, . . . correspond to sources; Z, Y, X, W correspond to documents containing copied text. Which document in a pair contains original text and which contains its copy was decided manually based on submission dates and contents of papers.

Figure 8 shows a subgraph corresponding to two particularly egregious cases of plagiarism. X is a PhD thesis, two thirds of which is copied from a variety of sources. Some sources are acknowledged, but most are not. Y is a journal article reviewing some area of physics. Several pieces of it are copied verbatim from other articles. It is an amusing coincidence that X and Y are connected: X copied in turn from Y.

Another node, A, has several adjacent edges for an opposite reason: A corresponds to lecture notes, and several people found it useful for their work. Apart from X which contains many pages stolen from A, another thesis, Z, shows a case of “mini-plagiarism”: a small chunk of 7 sentences is copied from A.

The overlap graph was not really necessary to discover X and Y: they had copied sufficiently many pages, even from single sources, that they had appeared in the top cases in the list of pairwise overlaps. Figure 9, on the other hand, shows a less obvious case of plagiarism that is easily visualized in the graphical representation. Node W here corresponds to an article comparing different methods, and descriptions of the methods are copied from other articles. Each description is short, so the separate cases in the list of pairwise overlaps might not appear to be significant. Combined together, however, they indicate that a large part of the article’s text is copied.

4. Related work

There has been much prior work on plagiarism detection tools. Most previous work was either intended for and tested on small sets of documents, often less than 100, or was designed and used for different types of text, such as programming assignments. There has been less work on scalable systems tuned for large text document collections.

Schleimer, Wilkerson, and Aiken [11] invented the winnowing algorithm, a variation of which we use here. It provides scalability and good efficiency, but since it was developed for Moss — a tool for checking programming assignments — it needed to be adapted for use on research articles. A significant number of additional pre- and postprocessing heuristics were required for application to the arXiv document collection.

Similarly, Broder [2] used document fingerprints, but chose the smallest k-gram hashes from the entire document, not from smaller windows. This permits detection of overall similarity between documents for duplicate detection, but not smaller overlaps between documents.

Brin et al. [1] developed COPS in 1994, a system designed to detect copying in research articles. COPS is based on strict comparison of sentences and therefore misses smaller modifications of text. It was also reported to have problems with parsing boundaries of sentences correctly. The approach is otherwise similar to ours: they developed a set of heuristics, such as elimination of common sentences, to focus on the most significant cases. Their small dataset of just 90 articles, however, did not permit development of further improvements required for processing large collections such as arXiv.

Koppel and Schler [6] suggested an approach to authorship detection, based on gradually removing the most useful features of a text and comparing with other documents using only the remaining features. Although the task is related, this approach cannot be directly applied to plagiarism detection because it assumes that documents are written by the single author, while we are interested in small plagiarized chunks of text.

In CHECK [14], another plagiarism detection system for text documents, Si et al. address the problem of complexity of pairwise comparisons by introducing hierarchical comparison: first only a set of primary keywords are used to compare documents, and more detailed comparison follows only if there is similarity at this top level. Comparison of each pair of files is still necessary, however, even if quite efficient for some of them. But a primary assumption of the authors, that “comparisons between documents addressing different subjects are usually not necessary in copy detection” does not hold in practice: we have discovered a number of cases in which for example a physics article includes description of a mathematical method copied from a mathematics article.

Collberg et al.’s system SPLAT [3] crawls the websites of top CS departments and collects research articles as a dataset aimed at detecting self-plagiarism. SPLAT works at the sentence level, with the similarity of documents based on the numbers of both similar and identical sentences, and where similarity of sentences is determined from the size of the intersection of their sets of words. While the approach works well for detecting self-plagiarism, it requires pairwise comparisons of documents and is thus not scalable.

Ribler and Abrams [10] suggested a method to visualize the degree of overlap of one document with a set of documents. All possible k-grams (not a winnowed subset) are collected from all documents, and k-grams of the document in question are plotted at different levels depending on how common it is. This method does not require pairwise document comparison, but requires collecting and keeping all k-grams, which would be an issue in large data sets. While indicating the presence of copied text, it does not identify its precise source.

The SCAM system [12] developed by Shivakumar and Garcia-Molina relies on word level analysis. Tuned to discover small overlaps, it results in many false positives when word distributions are similar but the texts are still different. This can happen often in research articles on the same subject. In later experiments with SCAM [13], the results are improved by eliminating the most common words, and by shifting to use of (unwinnowed) k-grams.

The approach used in MatchDetectReveal system by Monostori et al. [7] avoids using a hash-function due to concern about hash collisions. Instead they use algorithms for exact string comparison: each document is represented by a suffix tree data structure, without any loss of information, and is then compared with other documents represented as strings of texts. This approach provides the most exact results, but requires time consuming pairwise comparison. The authors suggest dealing with this problem by running comparisons in parallel on different cluster nodes.

In recent work, David and Pinch [4] used a modified version of our software to examine the extent and goals of copying and plagiarism in user reviews on amazon.com.

5. Conclusion

From a set of 284,834 docs covering a time period of over a decade, we found over 30,000 that might be considered duplicates, or excessive self-plagiarism: articles with overlapping authorship whose largest connected copy-free component was less than 20% of the total. We also identified over 500 cases of likely plagiarism from other authors, and additionally over 1000 cases of likely mild plagiarism. These constitute roughly 0.5% of the corpus, and an even smaller percentage of authors, since many come from repeat offenders. Some of the problems are quite serious, and many of the articles are published in conventional venues. None of the plagiarizers, the victims, nor their publishers have yet been notified, hence they remain anonymized here. We can, however, dispel some uncertainty by pointing out that while prominent (highly cited) authors are frequently victimized, they do not appear to reuse text from others.

The above results may tend to exaggerate the extent of the problematic behavior, but this needs further study. Many of the isolated copied sentences are of “background” nature, containing neither particularly unique information content nor stylistic virtue, and the “victims” might not even feel victimized. Some cases may reflect demographic and educational differences in an international author pool, with some careless reuse by non-native English writers who fear garbling content by modifying it. A next step underway is a private interface that displays the pairs of documents side by side, with overlapping text highlighted, and the ability to solicit confidential feedback from authors of both documents regarding the significance of the overlap.

We can also envision a module for real-time screening of new submissions. The calculation of fingerprints for an incoming document is computationally efficient, and the winnowed fingerprint hashes for the existing corpus fit in a couple gigabytes of memory, so the database lookups are also efficient. A flagged submission might receive the instant response "Overlap detected with article X by (same|different) authors, do you really wish to submit this?", with the effect of modifying the long-term behavior of authors. This would also assess how much better the honor system performs once the probability of detection is known to be high.

6. Acknowledgements

We thank Patrick Ng for his contributions to the data cleaning code. We also thank the Cornell Theory Center for usage of their equipment to run some of the experiments in this paper. This research was funded by NSF grants IIS-0636259, IIS-0541507, and by the KD-D Initiative. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

References

[1] S. Brin, J. Davis, and H. Garcia-Molina. Copy Detection Mechanisms for Digital Documents. In *Proceedings of the ACM SIGMOD Annual Conference*, May 1995.

[2] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29, 1997.

[3] C. Collberg, S. Kobourov, J. Louie, and T. Slattery. SPlaT: A System for Self-Plagiarism Detection. In *Proceedings of IADIS International Conference WWW/INTERNET 2003*, November 2003.

[4] S. David and T. Pinch. Six degrees of reputation: The use and abuse of online review and recommendation systems. *First Monday*, 11(3), 2006.

[5] P. Ginsparg. First steps towards electronic research communication. *Computers in Physics*, 8(4):390–396, 1994.

[6] M. Koppel and J. Schler. Authorship Verification as a One-Class Classification Problem. In *Proceedings of 21st International Conference on Machine Learning*, pages 489–495, July 2004.

[7] K. Monostori, A. Zaslavsky, and H. Schmidt. MatchDetectReveal: Finding Overlapping and Similar Digital Documents. In *Information Resources Management Association International Conference (IRMA2000)*, pages 955–957, May 2000.

[8] G. Ovtcharov, R. Dorsch, and M. Kruk. PDFTOHTML v0.36: A utility which converts PDF file into HTML and XML formats. Based on Xpdf 2.02 by Derek B. Noonburg. http://pdftohtml.sourceforge.net/ [Accessed 4 July 2006].

[9] M. O. Rabin. Fingerprinting by random polynomials. Technical report, Center for Research in Computing Technology, Harvard University, 1981.

[10] R. L. Ribler and M. Abrams. Using visualization to detect plagiarism in computer science classes. In *Proceedings of IEEE Symposium. on Information Visualization*, pages 173–178, October 2000.

[11] S. Schleimer, D. Wilkerson, and A. Aiken. Winnowing: Local Algorithms for Document Fingerprinting. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 76–85, June 2003.

[12] N. Shivakumar and H. Garcia-Molina. SCAM: A Copy Detection Mechanism for Digital Documents. In *Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries (DL'95)*, June 1995.

[13] N. Shivakumar and H. Garcia-Molina. Building a Scalable and Accurate Copy Detection Mechanism. In *Proceedings of 1st ACM International Conference on Digital Libraries (DL'96)*, March 1996.

[14] A. Si, H. V. Leong, and R. W. Lau. CHECK: A Document Plagiarism Detection System. In *Proceedings of ACM Symposium for Applied Computing*, pages 70–77, February 1997.

[15] S. Warner. The arXiv: Fourteen Years of Open Access Scholarly Communication. In M. Halbert, editor, *Free Culture and the Digital Library Symposium Proceedings*, pages 56–68, October 2005.