Duplicate News Story Detection Revisited

Omar Alonso
Microsoft Corporation
omalonso@microsoft.com

Dennis Fetterly
Microsoft Research, Silicon Valley Lab
fetterly@microsoft.com

Mark Manasse
Microsoft Research, Silicon Valley Lab
manasse@microsoft.com

Microsoft Research Technical Report
MSR-TR-2013-60

Abstract

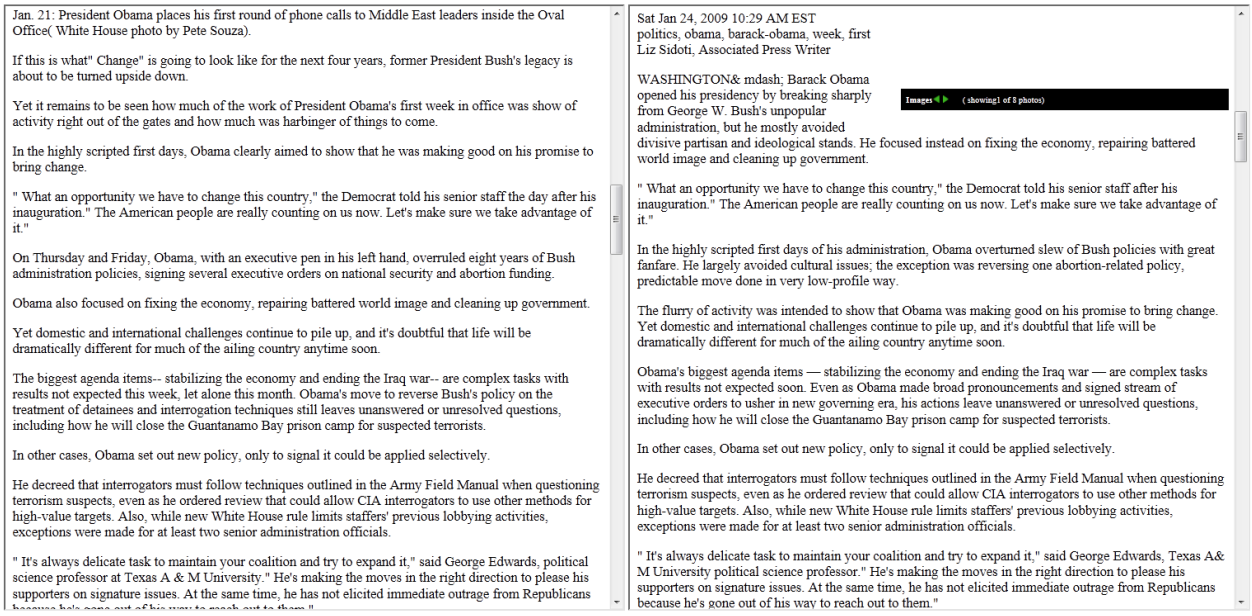
In this paper, we investigate near-duplicate detection, particularly looking at the detection of evolving news stories. These stories often consist primarily of syndicated information, with local replacement of headlines, captions, and the addition of locally-relevant content. By detecting near-duplicates, we can offer users only those stories with content materially different from previously-viewed versions of the story. We expand on previous work that improves the performance of near-duplicate document detection by weighting the phrases in a sliding window based on the term frequency within the document of terms in that window and inverse document frequency of those phrases. We experiment on a subset of a publicly available web collection that is comprised solely of documents from news web sites. News articles are particularly challenging due to the prevalence of syndicated articles, where very similar articles are run with different headlines and surrounded by different HTML markup and site templates. We evaluate these algorithmic weightings using human judgments to evaluate similarity. We find that our techniques outperform the state of the art with statistical significance and are more discriminating when faced with a diverse collection of documents.

1 Introduction

Near-duplicate document detection is a problem with a long history spanning many applications. Notable applications include duplicate result suppression for web search engines, filesystem compression through better dictionary seeding, reducing backup storage requirements, improved bandwidth utilization using delta compression from similar packets, copyright infringement, plagiarism detection, and visual image clustering by recognizing rotation, scale, and lighting invariants. Near-duplicate documents are those where the documents are not identical, so a hash-based comparison of the full content will fail, but are comprised of many identical features. Near-duplication is not necessarily transitive, $a \approx b$ and $b \approx c$ do not guarantee $a \approx c$.

In 2008, Theobald *et al.* investigated a set of techniques, *SpotSigs*, for detecting near-duplicate news items. Inspired by this, and armed with a relatively recent tool for approximating weighted Jaccard values for arbitrary non-negative weights, we seek to find algorithmic weightings (based solely on statistical properties of the collection) that yield comparable or better results, without the *ad-hoc* explicit choice of a small preferred set of stop words, as used by SpotSigs. In this context, we are concerned with the lexical similarity of the text, and are not aiming to identify semantically identical text fragments such as "The Greatest" and "Muhammad Ali", or "Venus" and "Morning (or Evening) Star"'; various techniques are known to work for semantic equivalence.

In the context of a web search service, algorithms for near-duplicate document detection face a number of real-world challenges, such as pages containing common navigational text, legal notices, user-generated content, contents of form fields (including lists of months and days), and content from services that suggest related pages or articles. These challenges are even greater in collections of news articles, due to syndication,



Jan. 21: President Obama places his first round of phone calls to Middle East leaders inside the Oval Office(White House photo by Pete Souza).

If this is what" Change" is going to look like for the next four years, former President Bush's legacy is about to be turned upside down.

Yet it remains to be seen how much of the work of President Obama's first week in office was show of activity right out of the gates and how much was harbinger of things to come.

In the highly scripted first days, Obama clearly aimed to show that he was making good on his promise to bring change.

" What an opportunity we have to change this country," the Democrat told his senior staff the day after his inauguration." The American people are really counting on us now. Let's make sure we take advantage of it."

On Thursday and Friday, Obama, with an executive pen in his left hand, overruled eight years of Bush administration policies, signing several executive orders on national security and abortion funding.

Obama also focused on fixing the economy, repairing battered world image and cleaning up government.

Yet domestic and international challenges continue to pile up, and it's doubtful that life will be dramatically different for much of the ailing country anytime soon.

The biggest agenda items-- stabilizing the economy and ending the Iraq war-- are complex tasks with results not expected this week, let alone this month. Obama's move to reverse Bush's policy on the treatment of detainees and interrogation techniques still leaves unanswered or unresolved questions, including how he will close the Guantanamo Bay prison camp for suspected terrorists.

In other cases, Obama set out new policy, only to signal it could be applied selectively.

He decreed that interrogators must follow techniques outlined in the Army Field Manual when questioning terrorism suspects, even as he ordered review that could allow CIA interrogators to use other methods for high-value targets. Also, while new White House rule limits staffers' previous lobbying activities, exceptions were made for at least two senior administration officials.

" It's always delicate task to maintain your coalition and try to expand it," said George Edwards, political science professor at Texas A & M University." He's making the moves in the right direction to please his supporters on signature issues. At the same time, he has not elicited immediate outrage from Republicans

Sat Jan 24, 2009 10:29 AM EST
politics, obama, barack-obama, week, first
Liz Sidoti, Associated Press Writer

WASHINGTON& mdash; Barack Obama opened his presidency by breaking sharply from George W. Bush's unpopular administration, but he mostly avoided divisive partisan and ideological stands. He focused instead on fixing the economy, repairing battered world image and cleaning up government.

Images (showing1 of 5 photos)

" What an opportunity we have to change this country," the Democrat told his senior staff after his inauguration." The American people are really counting on us now. Let's make sure we take advantage of it."

In the highly scripted first days of his administration, Obama overturned slew of Bush policies with great fanfare. He largely avoided cultural issues; the exception was reversing one abortion-related policy, predictable move done in very low-profile way.

The flurry of activity was intended to show that Obama was making good on his promise to bring change. Yet domestic and international challenges continue to pile up, and it's doubtful that life will be dramatically different for much of the ailing country anytime soon.

Obama's biggest agenda items — stabilizing the economy and ending the Iraq war — are complex tasks with results not expected soon. Even as Obama made broad pronouncements and signed stream of executive orders to usher in new governing era, his actions leave unanswered or unresolved questions, including how he will close the Guantanamo Bay prison camp for suspected terrorists.

In other cases, Obama set out new policy, only to signal it could be applied selectively.

He decreed that interrogators must follow techniques outlined in the Army Field Manual when questioning terrorism suspects, even as he ordered review that could allow CIA interrogators to use other methods for high-value targets. Also, while new White House rule limits staffers' previous lobbying activities, exceptions were made for at least two senior administration officials.

" It's always delicate task to maintain your coalition and try to expand it," said George Edwards, Texas A& M University political science professor." He's making the moves in the right direction to please his supporters on signature issues. At the same time, he has not elicited immediate outrage from Republicans because he's gone out of his way to reach out to them."

Figure 1: Two news articles discussing the same topic assessed as near-duplicates by the authors.

markup from software used by the publisher to control layout, local event listings, neighborhood shoppers (free papers) with repeated content, and section summaries mostly containing a single article from the section's subject in addition to abstracts for a number of additional articles. As an example of these challenges, Figure 1 shows the text version of two near-duplicate news articles. Some of the paragraphs in these articles have syntactic similarity, while others have semantic similarity. Note the difference in the paragraph breaks and slight rewording of the text which make near-duplicate detection a very difficult problem.

To evaluate our techniques when facing these real-world challenges, we experiment on subsets of a publicly available web collection, as well as on the SpotSigs test set We evaluate our algorithmic weightings using human judgments (by both the authors, and those generated using crowdsourcing) to evaluate both similarity and relevance to the task of detecting duplicate news stories. All variants of our techniques which eliminate templates outperform (with statistical significance) the state of the art and are more discriminating when faced with a diverse collection of relevant documents.

The remainder of this paper is laid out as follows. In Section 2 we discuss related work, Section 3 describes our approach, and the experiments are described in Section 4. Finally, we conclude in Section 5.

2 Related work

Near-duplicate detection algorithms have been utilized in a variety of ways. Some of the more significant ones, which we do not consider further in this paper, include: plagiarism detection [11, 13, 21], sub-document level replication [6, 2, 7, 18], Winnowing [20], finding near-duplicate files in a local or remote filesystem [15, 24, 17, 22], and web crawling [16].

Our consideration of the Theobald, *et al.* SpotSigs paper [23] suggested to us that the primary aspect differentiating their technique from the sampling approaches they rejected (shingling [3] and SimHash [5]) is that the Theobald approach is highly selective in the choice of phrases from which to draw samples. While SimHash is easily tuned to offer weighted sampling, the individual samples are words, not phrases. Preferentially weighting stop words would result in frequency matching of occurrences of those words (or their immediate successors), which is unlikely to distinguish news stories from one another. Shingling, as originally considered, uses phrases, and allowed for integer phrase weighting.

Gollapudi and Panigrahy [9] found a weighted sampling technique running in time logarithmic to the weights, suitable for weights larger than some positive constant; Manasse *et al.* [14] presented an expected

constant-time sampling algorithm for arbitrary non-negative weights. Ioffe [12] subsequently improved this to a Monte Carlo randomized algorithm running in constant time.

Recently, Gibson *et al.* [8] considered the specific problem of news story de-duplication, principally using a text extractor to reduce news pages to just the story, followed by the use of known sketching algorithms. We also tried a news story extractor in some of our algorithms to try to restrict our attention to just the news content of the story. Our techniques improve on Gibson by using aspects of the feature selection suggested by SpotSigs.

3 Algorithmic Approach

From Theobald [23], we take the idea that, in news stories, phrases beginning with common words are more likely to be body text of an article than to be captions or headlines. As such, these phrases are likely to be preserved as articles get rewritten as a breaking news story evolves, or as articles transit from a wire service to appearance in a newspaper. We therefore investigate phrase weightings giving higher weight to phrases beginning with oft-used terms.

From Broder [3] , we take the efficiency of replacing exact Jaccard computation by sample-based approximation, allowing the identification of highly-similar pairs drawn from billions of documents. We additionally take the idea of *supershingling*, the computation of hashes of a collection of samples, so that collisions in supershingle values strongly indicate high levels of simlarity, so that we can detect such instances in nearly linear time.

From Henzinger's comparison of shingling to SimHash [10], we take seriously that unweighted shingling is inferior to the weighted term selection of SimHash.

Ioffe's work on consistent sampling [12] offers a Monte Carlo constant-time technique for approximating weighted Jaccard values, allowing us to use arbitrary non-negative weightings of phrases. We use information-retrieval standards, such as term and document frequency, and modifications of such weightings using logarithms and powers.

Reflecting on these, we seek weighting schemes offering heightened probability of selection to those phrases SpotSigs prefers, but allowing some probability of selection to all phrases. We seeki weights so we can use weight-proportional sampling to create compact sketches of identified news articles. In contrast to SpotSigs, compact sketches (and supershingles) allow significantly larger corpora to be considered. SpotSigs presents clever techniques for rapidly determining if the similarity measure of a pair of files exceeds some threshold, but offers little to avoid considering all pairs.

SpotSigs chooses a small set of *antecedent* words common in English. It then selects only those phrases beginning with an element of the antecedent set for comparison using unweighted Jaccard. SpotSigs chose a test collection starting from a small number of known news stories, clustering near-duplicates in this small collection, and evaluating recall and precision for variants of SpotSigs within this collection. In attempting to replicate the SpotSigs results, we discovered that their reported results test only among related stories, largely eliminating false positives.

We taker a less restricted collection of documents, hoping to identify news stories within a large pool of documents selected from known news source web sites. We use a probabilistic approximation to weighted Jaccard to identify likely near-duplicates in this collection. The weights are based on term and phrase frequencies. We do not explicitly choose a preferred set of antecedents, and generally give all phrases which do not appear to be boilerplate some positive weight.

To elaborate and unify the computations involved, all of the algorithms assign a weight to every phrase of a chosen target length. SpotSigs confounds this simplification by picking phrases which omit the antecedent words; we do not attempt to replicate this behavior in our algorithms. For SpotSigs this seems inessential, most phrases will begin with either "the" or "said" but not both. SpotSigs then assigns weight one to phrases beginning at the position of a word in the antecedent set, and weight zero to all other phrases.

Weighted Jaccard for non-negative weightings W_1 and W_2 over a universe of phrases U is defined to be

$$\frac{\sum_{u \in U} \min(W_1(u), W_2(u))}{\sum_{u \in U} \max(W_1(u), W_2(u))}.$$

For binary weightings, the numerator is the cardinality of the intersection of W_1 and W_2, while the

denominator is the cardinality of the union of W_1 and W_2, viewing W_1 and W_2 as sets containing those elements with weight one, resulting in the conventional definition for the Jaccard value.

For whole-numbered weightings, the definition above corresponds to computing the unweighted Jaccard value of sets where each phrase u with weight $W(u)$ is replaced by new phrases $(u, 1), (u, 2), \ldots, (u, W(u))$.

In choosing weightings, in order to systematically approximate SpotSigs, we considered term frequency (both within a document, and in the entire corpus), and approximations or exact computation of phrase frequency in the corpus, in order to reduce the impact of boilerplate text.

Due to working with a corpus of considerable size, we call the one percent of phrases found in more than forty-two distinct documents *common*. This set can be efficiently stored in a relatively small amount of memory on even a modest computer and features that utilize both the presence and absence of a phrase in this set can be derived. For the same reason, we chose to work with unbiased estimations of the Jaccard value, rather than computing the exact Jaccard value for all pairs. The SpotSigs paper computes exactly the set of document pairs whose Jaccard value exceeds a chosen threshold, but it does this in a corpus where the number of pairs is bounded by a few million; we aim for corpora in which the number of individual documents is best measured in billions, resulting in quintillions of document pairs, rendering even the enumeration of all pairs impractical. We also seek to understand whether our algorithms help separate news stories from non-news, at least when one of the articles is judged as news. As such, we evaluated pairs across the spectrum of Jaccard similarity, rating selected pairs as irrelevant (two non-news stories), and duplicate or not. In our first experiments, as follows, we viewed the identification of a non-news story as a near-duplicate of a valid news story to be a false positive, reducing our precision value.

We discovered a few encouraging things: many variants of weighting produced results comparable to one another and to SpotSigs, as measured by F1 and Matthews values and by Pearson correlation of the results. We discovered many techniques using the document frequency of individual words to push the weighting towards a SpotSigs-like binary decision about the initial "stoppiness" of a phrase work about equally well: working better than SpotSigs on our new collection, but less well on the SpotSigs collection. We further discovered that SpotSigs performed surprisingly (to us) poorly on our broader collection, marking many documents as duplicates which shared only a significant amount of boilerplate text – for instance, the text associated with the navigational controls on different pages from a single news site – leading to low precision numbers for SpotSigs and for our first proposed weightings when applied to our test set.

We then refined some of the algorithms by down-weighting common phrases, using both a variant of inverse document frequency for phrases, and a simple threshold cut-off for phrases common to more than a few dozen documents. This improved the precision of our techniques to the point that our techniques significantly outperform SpotSigs on the judged portion of our collection. Using this variant to trim our sample collection, the F1 value of SpotSigs approached the values seen in their 2008 paper.

Computations using the algorithms described above are highly parallelizable – samples for different documents can be drawn independently in time linear in the document length multiplied by the number of samples to be drawn, if the phrase frequencies to determine weightings are in memory. Supershingling or other hashing techniques allow us to avoid the quadratic effort of considering all pairs of documents, allowing us to focus directly on pairs which collide, and therefore are likely to have sufficient similarity to be duplicates.

4 Evaluation

4.1 Experimental Setup

This section describes the data and computational infrastructure that we used to carry out our experiments. We started with the "Category A" English subset of the ClueWeb09 dataset[1] distributed by CMU. This subset contains 503 million English language web pages. Additionally, we retrieved the RDF version of the Open Directory Project site on September 23, 2010. In order to build a large test corpus comprised primarily of news documents, we filter the ClueWeb Category A English documents to contain only documents from one of the 7,261 distinct hostnames in the ODP News category, resulting in a set of 11,826,611 web pages. We further filter this set to remove exact duplicate pages by including only one representative from each

[1] `http://boston.lti.cs.cmu.edu/Data/clueweb09/`

Label	Frequency
Duplicate	42
Containment	10
Non-duplicate	252
Duplicate, Irrelevant	10
Non-duplicate, Irrelevant	142

Table 1: Distribution of labels for CW1.

group of exactly matching text pages. We also remove all content from Wikipedia. The resultiing collection of 5,540,370 web pages from news sites forms the corpus we use hereinafter.

To evaluate the effectiveness of near-duplicate detection algorithms, we first need to compute the similarity of the documents in the collection. It is computationally infeasible to compute the pairwise similarity of all 5.5 million documents. Past work [8, 23] has built small collections of documents either via clustering or by querying a search engine with an existing news article's title and retrieving the results to obtain duplicate stories. We took a different approach, which we believe yields a collection that surfaces many of the thorny issues in near-duplicate document detection.

We begin by extracting the potential news article from each of the documents in our corpus using the Maximum Subsequence Segmentation approach described in [19]. We then parse the documents using an HTML parser producing a sample of document pairs where the articles share at least one seven word phrase whose IDF is in the interval $[0.2, 0.85]$. From each group of documents that share a phrase, samples are drawn uniformly for all pairs in that group. The number of drawn samples is proportional to the number of pairs in the group. Once we have obtained the distinct set of pairs from all groups, we compute the unweighted Jaccard coefficient of the pairs of extracted articles. A histogram of these Jaccard values was then calculated and used to obtain a sample of 456 document pairs distributed approximately evenly across the set of Jaccard values.

Two authors labeled all of these pairs of pages, assigning a label with potential values of `Containment`, `Duplicate`, `Non-duplicate`, `Duplicate Irrelevant`, and `Non-duplicate Irrelevant`. The two sets of labels were compared for agreement, and the pairs where the labels were not in agreement were rejudged in consultation in order to obtain a set of labels with complete agreement. We refer to this dataset as CW1 for the remainder of this paper. The distribution of labels is depicted in Table 1. The labels are available from the authors via email.

It is worth noting that taking a random sample of pairs does not lead to a viable corpus of near-duplicate documents. To verify this, we selected 100 million document pairs at random. We then computed the Jaccard coefficient of the extracted article for each of these pairs. As expected, $99,932,445$ of the 100 million pairs had a Jaccard coefficient of 0 and $7,779$ had a coefficient of 1. The coefficients for the remaining pairs were distributed between 0 and 1, heavily skewed towards low values, so that labeling only these pairs does not produce a viable corpus.

In addition to the author-labeled near-duplicate news articles in ClueWeb09, we also used a set of crowdsourced labels described in the following subsection, as well as the Goldset[2] of documents used in [23]. The SpotSigs Goldset dataset has 2,167 articles covering 68 different news stories.

4.2 Crowdsourced Labels

Like many others, when attempting to scale the labeled dataset, we turned to crowdsourcing, where tasks are outsourced to an unknown set of workers. Using the same methodology as for CW1, we initially sampled 4,107 pairs of documents from our corpus. While labeling CW1, the authors used a labeling tool (implemented specifically for this task) that presented both documents at the same time, and assigned a label to the pair. This assignment actually incorporated multiple decisions: are both documents news articles, if so, are they about the same story, and finally, if they are about the same story, does one of them contain the other. After assessing CW1 and discussing disagreements, the authors defined some rules about what makes (or not) a news article. This categorization, along with several examples, was added to new versions of the

[2]Available from `http://www.mpi-inf.mpg.de/~mtb/`

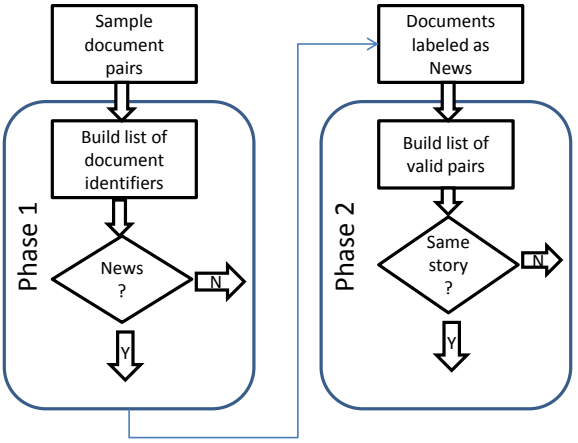


Figure 2: Flowchart of crowdsourcing pipeline used to construct CW2.

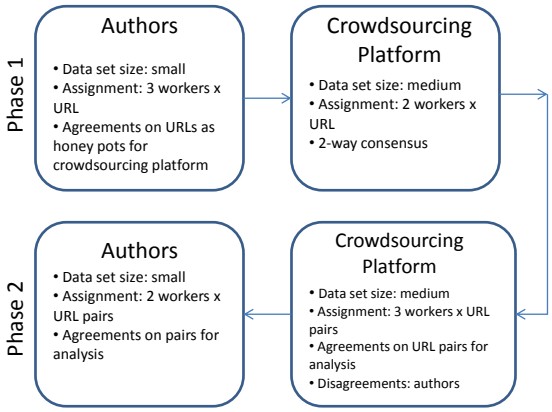


Figure 3: Flowchart of work quality control for phases 1 and 2.

experiments as part of the crowdsourcing pipeline. We streamlined this process for the labels we generated via crowdsourcing, which includes using a more generic tool for implementing the tasks.

To validate the new experimental design, we took the CW1 data set and ran it through phase 1 of the crowdsourcing pipeline. We compared the agreement between us and the workers using Cohen's Kappa and reported $\kappa = 0.766$, which indicates substantial agreement.

At a high level, we follow the same process described in [1] by using an iterative approach for the design and implementation of each experiment. We designed and tested both designs (Figures 4 and 5) with small data sets before involving crowds. We batched the data sets and adjusted quality control using honey pots and manually checking for outliers. We now describe each step in more detail.

We used a Microsoft internal system (Universal Human Relevance System) for gathering all crowd assessments.

We designed a crowdsourced labeling pipeline that consisted of two separate experiments: news identification (phase 1) and duplicate detection assessment (phase 2). The labeling process works as follows: Once we had our initial sample, we generated a list of distinct documents from the sampled pairs, and then asked workers to determine if a document was or was not a news article, or if the worker was unable to determine. After computing agreement, we then filtered the list of document pairs to include only those where both documents had been labeled as news articles. We then asked assessors if the articles in the pair were about the same event, and if so, if one of the articles had more detail than the other. Once there is output from phase 1, this process allows us to parallelize the assessment work by having the two experiments running at the same time. Figure 2 provides a graphical depiction of this process.

Work quality control is a key part in any task that requires human computation and we implemented a workflow that combines different crowds at different stages of both phases. Initially, we use a small data set to test the design of phase 1 experiments. Each URL was assessed by 3 workers (the authors) and all disagreements resolved in person. The output was used as a honey-pot data set to check the quality of the same experiment template using a different crowd (MS tool workers). In this step, an overlapping medium size data set is used and each URL is assessed by 2 workers. All those URLs where both workers agree are then used to generate the URL pairs, which are the input for phase 2. In this second phase, each URL pair was assessed by 3 workers and for those few cases where at least 2 of the additional workers disagree with the initial judgment, we (the authors) provided an extra label to compute the final list. Figure 3 describes the quality control mechanisms used in our work

We assessed 4,107 pairs in phase one, containing 3,992 distinct ids. Each document was assessed as a news article or not by two assessors where the values were one of `Yes`, `No`, `I don't know`, or `Other`. The assessments we received resulted in a Cohen's Kappa $\kappa = 0.73$, indicating substantial agreement. Table 2 lists the distribution of labels.

Once we had a set of documents that was labeled as news articles or not news articles, we returned to

Label	Frequency
Yes	4,235
No	7,877
I don't know	208
Other	412
Non English	45

Table 2: Distribution of raw labels for first crowdsourced task identifying documents that are news articles.

Please read the following document and let us know if the article is a news article.The following are considered news:

- Generic news articles (e.g., finance, politics, etc.)
- Press releases by a company or institution
- Government data reports or election returns

The following are not considered news:

- Weather reports (conditions)
- Headlines with a single or no sentences/paragraph or photo captions
- Event calendars
- Article or blog post that describes another article with a teaser quoting a small amount of content from the original article.
- Library collections or course listings

Does the document above contain a news article written in English?

- **Yes**. It is a news article.
- **No**. It is not a news article.
- **I don't know**
- **Other**. Web page didn't load/error message/etc.
- **Non English**. This document is not in English.

Figure 4: Experiment design for phase 1.

Please only consider the article itself (not the surrounding template). Ignore ads, images and formatting. We are only considering the core text of the articles. Note that headlines can be different.
1. Are these 2 news articles about the same event/topic?

- **Yes**. These news articles are about the same.
- **No**. These news articles are not the same.
- **I don't know**. I can't tell if the news articles are the same or not.
- **Other**. Web page didn't load/error message/etc.
- **Non English**. This document is not in English.

2. Does one document cover more detail than the other?

- Document A covers more detail than document B.
- Document B covers more detail than document A.
- No

Figure 5: Experiment design for phase 2.

the list of document pairs that we initially sampled and considered each pair where both documents had been labeled as news articles. The main goal of phase 2 was to answer two questions about each pair of news articles, as shown in Figure 5. Due to the experiment design, we expected all responses to question 1 to be either `yes` or `no`, we provided the other three responses as an escape clause if the judge was unsure of the result or to allow for operational issues. For phase 2, we report agreement using Fleiss' Kappa $\kappa = 0.74$, indicating again substantial agreement.

For the assessments from Phase 2, we directly take all URL pairs that have consensus and the authors resolved any disagreements by providing an additional assessment. Table 3 shows the results of this task. When comparing these results against CW1, note that Table 1 identifies duplicates and containment separately, while Table 3 counts all incidents of containment also as a duplicate. Therefore all of the results from Q2 - Yes are also counted in Q1 - Yes. Due to the two-phase nature of the crowdsourced pipeline, many sampled pairs were not carried forward from phase 1 to phase 2 because at least one article in the pair was not labeled as a news article. Despite the similar magnitude of the results inTable 1 and Table 3, the scale

Label	Frequency
Yes	311
No	265
I don't know	0
Other	0

Table 3: Distribution of raw labels for question 1 of second crowdsourced task identifying near-duplicate news articles.

uniform	$(\ln df)^2$	df^2
df	$(\ln df)^3$	df^3
$\ln df$	$(\ln df)^4$	df^4
	$(\ln df)^{10}$	$\ln(DocumentCount/df)$

Table 4: Functions to calculate final sample weight.

of the crowsourced experiment was much larger because of the labels obtained in phase 1. In this paper, we do not utilize the results from question 2; we leave this for future work. For the remainder of this paper, we refer to this dataset as CW2. As with CW1, the labels are available from the authors via email.

To conclude this section, it is important to describe operational issues that both crowds (authors and workers on UHRS) reported at different times. In general, with the exception of a few spammers, workers try to follow instructions as best they can. Unfortunately, at times the content is not in the best shape, creating problems for locating the article (i.e. "The news story is a way down the page, but is news"), server errors (i.e. "Half the page didn't load correctly and the bottom is just a photo is [sic] a caption"), or formatting problems (*i.e.* "is FCC report, but can't download it and in [sic] weird left-to right format... really not sure"). These issues do have an impact on the label quality so, by design, we selected those URLs where there was unanimous consensus before generating pairs.

4.3 Experimental Results

In order to compute sample values for our documents, we use Ioffe's [12] constant time weighted sampling technique.

Ioffe's technique extends on shingling, by selecting a weighted sketch of each document. We consider a document to be equivalent to its set of *phrases*, consecutive terms from the document. A weighted document is a document together with a weighting function, mapping each phrase to a non-negative weight. Symbolically, the set of phrases in a document is $\{\rho_i\}$, and a weight function maps each i to a non-negative value $W(i)$. A weighted sample $< \rho, w >$ is a pair where, for some i, $\rho = \rho_i$ and $0 < w < W(i)$. We want a family of sampling functions $\{\mathcal{F}_i\}$ where averaged over that family, the expected probability of agreement is equal to the Jaccard value. Thus, $Prob_i(\mathcal{F}_i(A) = \mathcal{F}_i(B)) = J(A, B)$. Such estimators are *unbiased*.

Ioffe shows that we can produce such a family by picking a family of pseudo-random uniform generators of values in the range between zero and one. For each phrase ρ, we seed the generator with ρ. We then compute five numbers $u_1, u_2, v_1, v_2, \beta$. Let $r = \frac{1}{u_1 u_2}$, and $t = \lfloor \beta + \log_r w \rfloor$. The weight y associated with phrase ρ is $r^{t-\beta}$, and the associated value $a = \frac{-\ln v_1 v_2}{ry}$. Looking at all phrases, choose as the sample the $< \rho, y >$ with the numerically least a value.

We experiment with several families of algorithms for setting weights. First, we generalize the SpotSigs approach of taking samples that occur immediately after one of a small number of antecedents. SpotSigs assigns equal weight to all antecedents. Variants of SpotSigs have antecedent sets that vary from the single term *is* to the 571 stopwords used in SMART [4]. We combine three different values to compute the weight for a given sample. First, we consider the document frequency of the first term in the window. Second, we scale this by the inverse document frequency of the complete phrase. Third, we multiply that by a binary value, which is 1 if the phrase is a "rare" phrase or 0 is the phrase is common. A rare phrase is a phrase that occurs in at most n% of the documents, as measured by document frequency. We computed results where the rare phrases are selected from the bottom 1%, 5%, 25%, 50%, 75%, 95%, and 99%. A function is then applied to the combined weight in order to determine a final weight for this phrase. Many of the values we utilize when calculating the weights are readily computed during the index construction phase for a search engine, or can be independently calculated with a pass over the corpus. The complete list of functions we applied is listed in Table 4. We use the following abbreviations when reporting results: DF (Document Frequency), IDF (Inverse Document Frequency), TP (True Positive), and FN (False Negative).

4.4 Results on ClueWeb Labeled Data

As described in Section 4.1, we drew plausible pairings by examining the Jaccard similarity of paired pages. We then took samples of roughly equal size from small ranges of Jaccard similarity, so that our samples would span the gamut of syntactic similarity. We chose the uniform distribution so that we could explore our techniques in a variety of settings.

All sampled document pairs were split apart, and individual pages examined to check whether the pages were news stories. For those pairs consisting of two news stories, we then asked our judges two further questions: Were the stories about the same thing? Did one story subsume the other? These questions determined whether the pages were news duplicates, and whether one contained the other.

We performed the experiments twice, drawing independent samples from the same collection. We then computed the Earth Mover Distance between the resulting distributions of documents. The first set, which we judged entirely in-house, had a distribution very close to uniform, although slightly oversupplied with low Jaccard similarity when compared to the uniform distribution. The second distribution is somewhat different; it was roughly three times farther from the uniform distribution than was our first sample. The Jaccard distribution of the first sample was significantly closer to the uniform distribution than it was to the new sample distribution. We do not fully understand the genesis of these discrepancies; one possible explanation is that the second samples were drawn precisely as described above, while the first set of samples were concurrently assessed as pairs for duplication as well as newsworthiness. We suspect that the difference is due in part to the level of personal investment by judges; when asked first to decide whether a page contains news, the judge has no personal stake in the page as yet. When asked to first decide whether two pages are duplicates, and then asked if they are news, the judge has to think more deeply as to whether the similar aspects of the pages constitute news, focusing the attention of the judge on the parts of the page they deem similar. More simply, the sets of pairs were drawn before ascertaining relevance to the task, but the surviving pairs in the later experiment are only those deemed to be news, reducing the set of pairs by eliminating many non-textual pages.

To assess the quality of our techniques for automatically determing when two articles are dupiicates in the desired sense, we look at the F1 measure. Because our functions are not linearly-related, the threshold values we use for cutoffs are largely unrelated. Accordingly, we set the threshold for each technique by choosing a sample of the document pairs, and finding the threshold value resulting in the maximum F1 score on this set. We then use this threshold on the full set of pairs and compute and report the F1 scores. Because of the differing scales for each technique, we then compare the F1 scores for thresholds in small neighborhoods of the predetermined threshold, to assess sensitivity of our techniques to the choice of threshold value.

The F1 measure is easily described in terms of recall and precision, which measure the fraction of detected positives, $\frac{TP}{TP+FN}$, and the fraction of positives which are correct, $\frac{TP}{TP+FP}$. Using these, F1 is

$$2\frac{recall \times precision}{recall + precision} = \frac{2TP}{2TP + FN + FP}$$

.

In a particular example we considered weights equal to the fourth power of document frequency for the first word in a phrase, divided by the document frequency of the entire phrase. We computed recall, precision, and F1 curves at a range of thresholds from zero to one. In this case, the curve stayed reasonably flat from thresholds of $\frac{1}{4}$ to $\frac{3}{4}$, suggesting relative insensitivity to the precise setting of the threshold.

All of the experiments in this paper use a phrase length of 7. SpotSigs does not consider phrase windows of a fixed size, instead, "chains" of some chain length c are constructed where the the non-stopwords that are at least d terms apart are selected. The best settings reported in the SpotSigs paper are a chain length of 3 plus a distance of 2, akin to our phrase length of 7 in the absence of stopwords within the phrase window: we include the first term in the phase window while SpotSigs omits it.

4.5 Rare Phrases

We also experimented with considering various fractions of rare (or not-so-rare) phrases. We calculated results for all of the functions in Table 4 where we set the weight for a phrase to 0 if it occurred more frequently than the bottom 1, 5, 25, 50, 75, 95, and 99% of phrases by document frequency. For the ClueWeb09 dataset, this table of rare phrases contains at most 847,281 elements, which can be easily stored

Antecedent list	Max F1
Is	0.6720
The	0.7868
Is,The	0.7851
Is,The,Said	0.7901
Is,The,Said,Was	0.7896
Is,The,Said,Was,There	0.7886
Is,The,Said,Was,There,A	0.7918
Is,The,Said,Was,There,A,It	0.7967
The,A,Can,Be,Will,Have,Do	0.7734
Smart stopword list	0.7572

Table 5: Maximum F1 scores for SpotSigs similarity.

Weighting function	Max F1
uniform	0.8352
df	0.8275
$\log DF$	0.8429
$\log Idf$	0.8093
$\log DF^2$	0.8505
$\log DF^3$	0.8449
$\log DF^4$	0.8470
$(\log DF)^2$	0.8352
$(\log DF)^3$	0.8288
$(\log DF)^4$	0.8257
$(\log DF)^10$	0.8307

Table 6: Maximum F1 scores for rare phrase weighting functions.

with the counts in an in-memory hashtable (we did not, but suspect one could, use a streaming heavy hitters algorithm to probabilistically identify common phrases.) While all of the rare phrase variants of our technique perform well, we find that considering the bottom 50% performs best. Table 6 lists the F1 value we compute for each weighted sampling technique. The F1 values we computed for all rare phrase variants are superior to all other techniques, including SpotSigs, which are listed in Table 5, primarily (as best we can guess) due to the presence of significant amounts of boilerplate in newspaper formatting, which has little to do with the contents of an individual story.

We compared SpotSigs similarity against Rare Phrase similarity by looking at all of our pairs of documents. We scored each algorithm by whether it correctly predicted the judged assessment of similarity. Looking at just the judgements, all of our algorithms compared to the best SpotSigs algorithm produced a large number of points of agreement with the judges and with one another. Simple χ^2 testing revealed no significant differences: both disagreed about equally often with each often choosing positive.

However, as Table 6 shows, our F1 scores are typically 5% better than SpotSigs, and, when considering whether we agree with the judges at points of disagreement, the likelihood of that improved F1 score being due to chance is almost always below 2% as measured by χ^2, and often vanishingly small as measured using a two-sided T test.

4.6 Pearson Correlation

We computed Pearson's sample correlation coefficient, r, for all methods against the truth set of labels. This coefficient is in the range $[-1, 1]$ where a value of 0 signifies no correlation between the variables and a value of 1 signifies that the variables are perfectly correlated; the Pearson correlation coefficient is the best linear term in mapping two real variables after normalizing for scale. The top five weighting functions for Pearson correlation coefficient are listed in Table 7. As depicted by the table, we see that near duplicate detection using weighted samples is highly correlated with human labels of duplicate news documents. It is worth noting that the best functions for weights all incorporate the least popular 50% of phrases, as further evidence that this variant is the most selective. Several of the weighting functions with the lowest correlation coefficients were functions with uniform weights, which is very close to Shingling. In terms of comparison with SpotSigs, the best SpotSigs correlation coefficient is 0.5517, which is for the variant where the antecedent set consists of: `Is, The, Said, Was, There`.

In computing Pearson correlation to the truth set, we mapped `False` and `True` to zero and one respectively (any distinct constants would do, due to scale invariance).

We also used Pearson correlation to check the similarity between different choices of method. Setting a threshold for Pearson correlation of at least 0.9, we find that all of the SpotSigs variants other than `The` correlate with one another. For weighted sampling, within blocks of similar weighting functions we find strong correlation. The correlation holds completely at the 0.9 level among all rare phrase variants at the 1 and 5% level, and at the 25 and 50% level. Reducing the cutoff to 0.8 causes most of the inverse document

Weighting function	Pearson r
df^2 rare phrase 50%	0.7564
df^3 rare phrase 50%	0.7550
df^4 rare phrase 50%	0.7530
df rare phrase 50%	0.7447
$(\ln df)^{10}$ rare phrase 50%	0.7437

Table 7: Top 5 weighting functions for Pearson correlation coefficient.

frequency and first-term weight variants to closely resemble SpotSigs variants. It also causes almost all rare phrase variants at 25% to match all other rare phrase variants.

4.7 Matthews Correlation

The Matthews Correlation Coefficient, or MCC, which is defined to be

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

is a correlation coefficient with a value in the range $[-1, 1]$ commonly used to evaluate machine learning algorithms. A correlation coefficient value of 1 indicates perfect classification. Figure 6 shows the MCC for a select few of the methods we experimented with. Each of the displayed methods is compared to the ground truth. We selected the best-performing methods within each class of technique; other methods performed similarly.

We observe that our rare phrase variants do quite well compared to the best variant of SpotSigs, except for a small range of thresholds. We consider this acceptable: the variants we chose concentrate around a threshold of roughly $\frac{1}{4}$ to $\frac{1}{2}$. There is no intrinsic correlation between the thresholds for different methods; it makes sense to select a value for each that typically performs well.

Again, the worst-performing methods are a uniform weighting (which closely approximates shingling), and the best variant of SpotSigs. As noted by Henzinger, straight shingling is an inferior method, falling prey to a host of irrelevancies in the document. The inverse phrase-frequency weighting has intermediate performance: better than SpotSigs, roughly equal to uniform rare phrase (but with a flatter range for tuning), and inferior to an exponentially-weighted rare phrase variant. Although not depicted, these comparisons are consistent across the gamut of variants.

4.8 Results on SpotSigs GoldSet

We computed the SpotSigs variants as well as the weighted sampling measures evaluated over the 2,346,862 document pairs that exist in the SpotSigs GoldSet, where ground truth values are determined by the directory hierarchy. On the GoldSet, we observe that some SpotSigs variants outperform the weighted sampling techniques. SpotSigs configured to use an antecedent list of [Is, The, Said, Was] performs the best with an F1 score of 0.87, while the best weighted sampling function, df^4 configured using rare phrases, obtains a F1 score of 0.84. There may be multiple factors that contribute to this outcome. The GoldSet set is relatively small, with only 68 clusters of articles. Hence many documents that contain specific phrases are indicative of these clusters, which is unlike the larger ClueWeb collection, where duplicate phrases occur in non-duplicate documents very frequently due to legal notifications such as privacy policies, instructions for comment areas, and the shared boilerplate that comes from a small number of corporations owning a large number of publications. Further investigation is required to fully determine the reasons for this difference. We note only that the existing SpotSigs algorithms perform comparatively poorly on our other dataset.

5 Conclusion

We have presented an algorithm for effectively detecting near duplicate news stories. This algorithm generalizes the SpotSigs approach by using the term and phrase frequencies to weight sample choices. Further, non

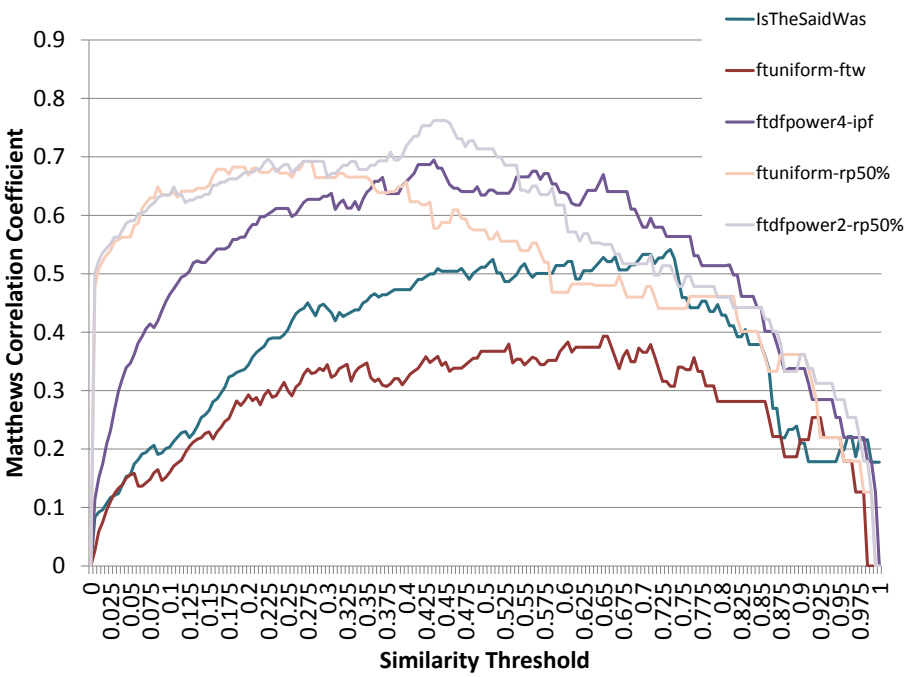


Figure 6: Matthews Correlation Coefficient for a selection of techniques.

binary weights are utilized giving our approach more of a SimHash flavor, thereby addressing issues raised by Henzinger. Our experimental results significantly improve performance using a battery of statistical tests on a test set presenting real-world challenges. In addition to the algorithm, we make our test set available for use by the research community.

In this work, we took note of only very-common phrases due to the difficulty of exact counting of frequency given the large number of uncommon phrases. In the future, approximate counting Bloom filters might be an economical way to find most of the heavy hitters. We had the opportunity to use exact counting, but given that we used frequency for rarity only as a binary decision, fuzzier counts would suffice.

While we may think that assessing duplicate documents is a simple task, in practice it is difficult and demanding. There are a number of presentation issues (e.g., formatting, broken images, fonts, colors, different styles, etc.) that the assessor has to deal with to locate the "core" of the document. Different news agencies often produce different paragraph breaks making it difficult for workers to find visual anchors to compare similarity. To make matters worse, paragraphs are also reordered. To address these issues we implemented a tool for cleaning up documents so it is possible to perform document assessment without formatting distractions.

We also introduced a crowdsourcing pipeline that consists of two phases for gathering labels and improves the overall label quality. As part of the quality control, the authors resolved many disagreements and even there we couldn't always agree and had to manually break ties.

Assessing archival or historical reference collections where part of the visual material is not available is a challenge as workers have to make an effort to locate the important pieces of material first, before producing any labels.

References

[1] O. Alonso. Implementing crowdsourcing-based relevance experimentation: an industrial perspective. *Information Retrieval*, pages 1–20. 10.1007/s10791-012-9204-1.

[2] M. Bendersky and W. B. Croft. Finding text reuse on the web. In R. A. Baeza-Yates, P. Boldi, B. A. Ribeiro-Neto, and B. B. Cambazoglu, editors, *WSDM*, pages 262–271. ACM, 2009.

[3] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In *Selected papers from the sixth international conference on World Wide Web*, pages 1157–1166, Essex, UK, 1997. Elsevier Science Publishers Ltd.

[4] C. Buckley, G. Salton, and J. Allan. Automatic retrieval with locality information using SMART. In *First Text Retrieval Conference (TREC-1)*, pages 69–72, 1992.

[5] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 380–388, New York, NY, USA, 2002. ACM.

[6] A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM TRANSACTIONS ON INFORMATION SYSTEMS*, 20(2):2002, 2002.

[7] D. Fetterly, M. Manasse, and M. Najork. Detecting phrase-level duplication on the world wide web. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 170–177, New York, NY, USA, 2005. ACM.

[8] J. Gibson, B. Wellner, and S. Lubar. Identification of duplicate news stories in web pages. In *Proceedings of the 4th Web as CorpusWorkshop (WAC-4)*, 2008.

[9] S. Gollapudi and R. Panigrahy. Exploiting asymmetry in hierarchical topic extraction. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 475–482, New York, NY, USA, 2006. ACM.

[10] M. Henzinger. 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, pages 284–291, New York, NY, USA, 2006. ACM.

[11] T. C. Hoad and J. Zobel. Methods for identifying versioned and plagiarized documents. *J. Am. Soc. Inf. Sci. Technol.*, 54(3):203–215, Feb. 2003.

[12] S. Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. *IEEE International Conference on Data Mining*, 0:246–255, 2010.

[13] W. Kienreich, M. Granitzer, V. Sabol, and W. Klieber. Plagiarism detection in large sets of press agency news articles. *Database and Expert Systems Applications, International Workshop on*, 0:181–188, 2006.

[14] M. Manasse, F. McSherry, and K. Talwar. Consistent weighted sampling. Technical Report MSR-TR-2010-73, Microsoft Research, 2010.

[15] U. Manber. Finding similar files in a large file system. In *Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference*, WTEC'94, pages 2–2, Berkeley, CA, USA, 1994. USENIX Association.

[16] G. S. Manku, A. Jain, and A. Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 141–150, New York, NY, USA, 2007. ACM.

[17] A. Muthitacharoen, B. Chen, and D. Mazières. A low-bandwidth network file system. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, pages 174–187, New York, NY, USA, 2001. ACM.

[18] M. Najork. Detecting quilted web pages at scale. In *Proceedings of the 35th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, New York, NY, USA, 2012. ACM.

[19] J. Pasternack and D. Roth. Extracting article text from the web with maximum subsequence segmentation. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 971–980, New York, NY, USA, 2009. ACM.

[20] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 76–85, New York, NY, USA, 2003. ACM.

[21] B. Stein, S. M. zu Eissen, and M. Potthast. Strategies for retrieving plagiarized documents. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 825–826, New York, NY, USA, 2007. ACM.

[22] D. Teodosiu, N. Bjørner, Y. Gurevich, M. Manasse, and J. Porkka. Optimizing file replication over limited-bandwidth networks using remote differential compression. Technical Report MSR-TR-2006-157, Microsoft Research, 2006.

[23] M. Theobald, J. Siddharth, and A. Paepcke. Spotsigs: robust and efficient near duplicate detection in large web collections. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 563–570, New York, NY, USA, 2008. ACM.

[24] A. Tridgell and P. Mackerras. The rsync algorithm. Technical Report TR-CS-96-05, Australian National University, Department of Computer Science, June 1996. (see also: http://rsync.samba.org).