Duplicate News Story Detection Revisited

Omar Alonso[1], Dennis Fetterly[2], and Mark Manasse[2]

[1] Microsoft Corporation
omalonso@microsoft.com
[2] Microsoft Research, Silicon Valley Lab
{fetterly, manasse}@microsoft.com

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 and improve 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 determine 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.

Keywords: News story near-duplicate detection, web similarity, crowdsourcing

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, better filesystem compression through careful dictionary seeding, reduced storage requirements for backups, improved bandwidth utilization using delta compression on similar packets, copyright infringement or plagiarism detection, and visual image clustering by recognizing rotation, scale, and lighting invariant features. 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 a plurality of identical features. Near-duplication is not necessarily transitive: $a \approx b$ and $b \approx c$ do not guarantee $a \approx c$.

In a web search service, algorithms for near-duplicate document detection face many 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, 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.

In 2008, Theobald *et al.* investigated a set of techniques, *SpotSigs*, for detecting near-duplicate news items. Inspired by this, and newly armed with a tool for approximating arbitrary non-negative weighted Jaccard values, we seek to find algorithmic weightings (based solely on the corpus' statistical properties) that yield comparable or better results, without the *ad-hoc* explicit choice of a preferred set of stop words, unlike SpotSigs.

To evaluate our techniques when facing these real-world challenges, we experiment on subsets of a publicly available web collection. We evaluate our algorithmic weightings using human judgments by the authors and from crowdsourcing to evaluate similarity and relevance when detecting duplicate news stories. All variants of our techniques which eliminate templates outperform (with statistical significance) previous efforts, with better discrimination in a more 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.

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, 22], sub-document replication [6, 2, 7, 18], Winnowing [21], finding near-duplicate files in a local or remote filesystem [15, 25, 17, 23], and web crawling [16].

Our consideration of the Theobald, *et al*. SpotSigs paper [24] suggested 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 (a technique for approximating cosine similarity) 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 cannot distinguish news stories from one another. Shingling, as described, used phrases, but 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 given 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. We omit this comparison due to space limatations.

3 Algorithmic Approach

From Theobald [24], we take the idea that phrases beginning with common words are more likely to be body text of an article than to be captions or headlines. These phrases are likely to be preserved as a 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 as measured by term frequency.

From Broder [3], we take the efficiency of replacing exact Jaccard computation by sample-based approximation, allowing the identification of most highly-similar pairs drawn from billions of documents.

From Henzinger's comparison of shingling to SimHash [10], we take seriously that unweighted shingling is often 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.

We seek weighting schemes offering heightened probability of selection to those phrases SpotSigs prefers, but allowing some probability of selection to all phrases. We use weight-proportional sampling to create compact document sketches. Unlike SpotSigs, compact sketches and supershingles allow significantly larger corpora to be considered.

SpotSigs chooses a small set of *antecedent* words common in English. It then selects samples beginning after an element of the antecedent set skipping antecedents and a configurable number of terms. SpotSigs chose a test collection starting from a small number of known news stories, building 68 clusters of near-duplicate articles, and evaluating recall and precision for variants of SpotSigs within this collection.

We 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; a behavior we did not attempt to replicate. SpotSigs 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

$$\frac{\sum_{u\in U}\min(W_1(u),W_2(u))}{\sum_{u\in U}\max(W_1(u),W_2(u))}=\frac{\|\min(W_1,W_2)\|_1}{\|\max(W_1,W_2)\|_1}.$$

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.

In choosing weightings, 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. Features that utilize both the presence and absence of a phrase in this set can be derived. We chose to work with unbiased estimates of the Jaccard value, rather than computing the exact 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 value and Matthews coefficient. 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 (IDF) for phrases, and a simple threshold cut-off for phrases common to more than a few dozen documents. The resulting techniques significantly outperform SpotSigs on the judged portion of our collection.

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.

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 503 million page "Category A" English subset of the ClueWeb09 dataset[3]. 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 pages to contain only those from one of the 7,261 distinct hostnames in the ODP News category, resulting

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

in a set of 11,826,611 web pages. We then removed all content from Wikipedia and exact duplicate pages by including only one representative from each group of exactly matching text pages. The resulting collection of 5,540,370 web pages forms our corpus.

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, 24] 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 samples drawn 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, along with the set of document identifiers that form our corpus, are available from the project page at `http://research.microsoft.com/projects/newsdupedetect/`.

Table 1. Distribution of labels for CW1.

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

Table 2. Distribution of Phase 1 raw labels.

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

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. 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 (Figure 3 depicts Phase 2) 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 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 asked workers if the articles in the pair were about the same event and if one had more detail than the other. The Microsoft Universal Human Relevance System [20] gathered all crowd assessments.

Quality control is a key part in any crowdsourcing task and our workflow combines different crowds at each phase. Initially, we use a small data set to test the design of phase 1 experiments. Each URL was assessed by 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 phase using a different crowd. In this step, an overlapping medium size data set is used and each URL is assessed by 2 workers. All 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 provided an extra label to compute the final list. Figure 2 describes the quality control mechanisms we used.

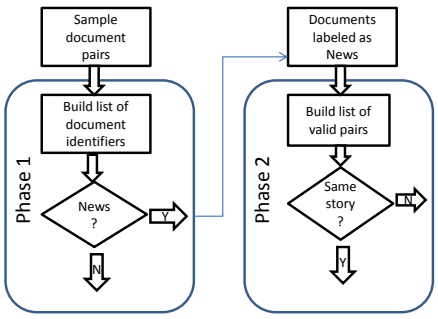

Fig. 1. Flowchart of crowdsourcing pipeline used to construct CW2.

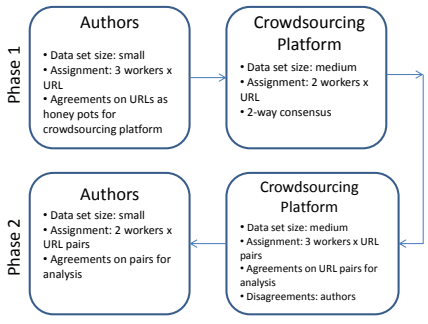

Fig. 2. Flowchart of work quality control for phases 1 and 2.

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 workers 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.

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

Fig. 3. Experiment design for Phase 2.

Once we had a set of documents that was labeled as news articles or not news articles, we returned to the list of document pairs that we initially sampled and considered each pair where both documents had been labeled as news articles. 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 assessed as news. 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. 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 in Tables 1 and 3, the scale of the crowdsourced experiment was much larger because of the labels obtained in Phase 1. We refer to this dataset as CW2 and these labels are also available from the project webpage.

4.3 Experimental Results

In order to compute sample values for our documents, we use Ioffe's [12] constant time weighted sampling technique, which extends shingling to select a weighted sketch [5] 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,

Table 3. Phase 2 label Distribution.

Label	Frequency
Yes	323
No	386
I don't know	0
Other	1

Table 4. Functions used for final weight.

uniform	$\log^2 DF$	DF^2
DF	$\log^3 DF$	DF^3
$\log DF$	$\log^4 DF$	DF^4
$\log(DocumentCount/DF)$	$\log^{10} DF$	

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 $0 \leq w < W(i)$. 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 produces such a family by picking a family of pseudo-random generators of values. For each phrase ρ, we seed the generator with ρ, and uniformly pick five numbers $u_1, u_2, v_1, v_2, \beta \in [0,1)$. 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 is $a = \frac{-\ln v_1 v_2}{ry}$. From 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 two values to compute the weight for a given sample. First, we consider the document frequency (DF) of the first term in the window. Second, we either scale this by the IDF of the complete phrase or multiply that by a binary value, which is 1 if the phrase is a "rare" phrase or 0 is the phrase is common, as detailed in Section 4.5. A function from Table 4 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.

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 took samples of roughly equal size from small ranges of 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.

We utilize the F1 measure to assess the quality of our techniques. 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.

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 SpotSigs results are for 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. Future work could consider other phrase lengths.

Table 5. Maximum F1 scores for SpotSigs similarity.

Antecedent list	Max F1
Is	0.7394
The	0.8375
Is,The	0.8321
Is,The,Said	0.8373
Is,The,Said,Was	0.8382
Is,The,Said,Was,There	0.8385
Is,The,Said,Was,There,A	0.8392
Is,The,Said,Was,There,A,It	0.8446
The,A,Can,Be,Will,Have,Do	0.8263
Smart stopword list	0.8202

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

Weighting function	Max F1
uniform	0.8693
DF	0.8625
$\log DF$	0.8732
$\log IDF$	0.8692
DF^2	0.8825
DF^3	0.8815
DF^4	0.8824
$\log^2 DF$	0.8698
$\log^3 DF$	0.8710
$\log^4 DF$	0.8776
$\log^{10} DF$	0.8807

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 in more than 1, 5, 25, 50, 75, 95, or 99% of documents. For the ClueWeb09 dataset, this table of rare phrases contains at most 847,281 elements, which can be easily stored with the counts in an in-memory hashtable. 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 at 50%. 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 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 judgments, 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 6% 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 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 4 shows the MCC for a select few of the methods we experimented with compared to the ground truth data from CW1. 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 (to save space), these comparisons are consistent across the gamut of variants.

5 Conclusion

We have presented an algorithm for effectively detecting near duplicate news stories. This algorithm generalizes SpotSigs by using the term and phrase frequencies to weight sample choices. Non-binary weights give our approach more of a SimHash flavor, 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. We make both our algorithm and our test set available for use by the research community.

We plan to test the suitability of this technique across multiple languages. By using DF as a term weight, we hope this will work even in languages without stopwords.

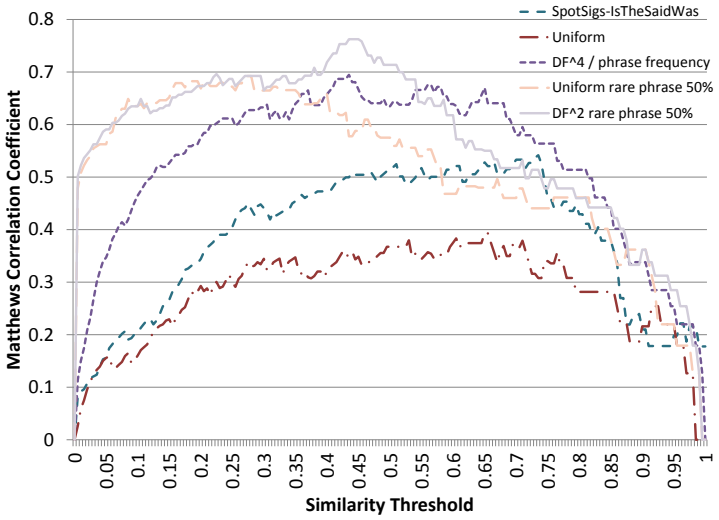

Fig. 4. Matthews Correlation Coefficient for a selection of techniques on CW1.

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, 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.

We also introduced a crowdsourcing pipeline that consists of two phases for gathering labels and improves the overall label quality. The two phase pipeline let us maximize the utility of our assessment effort, since many candidate pairs were dropped from consideration after Phase 1 after an element was assessed to be non-news. Initially sampling candidate documents and then for those judged to be news, pairing to documents with desired similarity as a three phase process might have increased the yield.

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.

6 Acknowledgments

We thank Jeff Pasternack for the use of his Maximum Subsequence Segmentation library (`http://www.jeffreypasternack.com/software.aspx`) for news article extraction.

References

1. O. Alonso. Implementing crowdsourcing-based relevance experimentation: an industrial perspective. *Information Retrieval*, pp. 1–20, 2012.
2. M. Bendersky and W. B. Croft. Finding text reuse on the web. In *WSDM*, pp. 262–271, 2009.
3. A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In *WWW*, pp. 1157–1166, 1997.
4. C. Buckley, G. Salton, and J. Allan. Automatic retrieval with locality information using SMART. In *TREC-1*, pp. 69–72, 1992.
5. M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *ACM STOC*, pp. 380–388, 2002.
6. A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM TOIS*, 20(2):2002, 2002.
7. D. Fetterly, M. Manasse, and M. Najork. Detecting phrase-level duplication on the world wide web. In *ACM SIGIR*, pp. 170–177, 2005.
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 *ACM CIKM*, pp. 475–482, 2006.
10. M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *ACM SIGIR*, pp. 284–291, 2006.
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 ℓ_1 sketching. *IEEE ICDM*, pp. 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*, 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 *USENIX WTEC*, Berkeley, 1994.
16. G. S. Manku, A. Jain, and A. Das Sarma. Detecting near-duplicates for web crawling. In *WWW*, pp. 141–150, 2007.
17. A. Muthitacharoen, B. Chen, and D. Mazières. A low-bandwidth network file system. In *ACM SOSP*, pp. 174–187, 2001.
18. M. Najork. Detecting quilted web pages at scale. In *ACM SIGIR*, 2012.
19. J. Pasternack and D. Roth. Extracting article text from the web with maximum subsequence segmentation. In *WWW*, pp. 971–980, 2009.
20. R. Patel. UHRS overview. `http://research.microsoft.com/en-us/um/redmond/events/fs2012/presentations/Rajesh_Patel.pdf`
21. S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In *ACM SIGMOD*, pp. 76–85, 2003.
22. B. Stein, S. M. zu Eissen, and M. Potthast. Strategies for retrieving plagiarized documents. In *ACM SIGIR*, pp. 825–826, 2007.
23. 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.
24. M. Theobald, J. Siddharth, and A. Paepcke. Spotsigs: robust and efficient near duplicate detection in large web collections. In *ACM SIGIR*, pp. 563–570, 2008.
25. A. Tridgell and P. Mackerras. The rsync algorithm. Technical Report TR-CS-96-05, Australian National University, Dept. of Computer Science, June 1996. (cf: `http://rsync.samba.org`).