# Large Scale Page-Based Book Similarity Clustering

Nemanja Spasojević
*Google Inc.*
*1600 Amphitheatre Parkway*
*Mountain View, CA 94043, USA*
*Email: sofra@google.com*

Guillaume Poncin
*Google Inc.*
*1600 Amphitheatre Parkway*
*Mountain View, CA 94043, USA*
*Email: gponcin@google.com*

***Abstract*—The Google Books corpus now counts over 15M books spanning 7 centuries and countless languages. Traditional cataloguing at that scale is imprecise, and often fails to identify more complex book-to-book relationships, such as 'same text, different pagination' or 'partial overlap'. Our contribution is a two-step technique for clustering books based on content similarity (at both book and page level) and classifying their relationships. We run this on our corpora consisting of more than 15M books (5B pages). We first detect similar books and similar pages within matching books, using hashing techniques and judicious thresholds. We then combine those features to identify the exact relationship between matching books. In this paper, we describe the basic approach to making the problem tractable, as well as the features and classifiers that we used. We enumerate a small number of relationships to qualify the link between scanned real-world books. Finally, we provide precision and recall measurements of the classifier.**

***Keywords*-Document similarity detection; relation classification; min-hash; book clustering;**

## I. Introduction

The last decade has seen the development of multiple mass digitization projects: *Live Search Books*, *Internet Archive*, *Google Books* to name only a few. The number of distinct books owned by libraries in the world is estimated to be around 130 million [1] and a large fraction is becoming available in digitized form. Because digital repositories gather content from multiple libraries, we expect significant overlap in the material scanned, even within collections of the same library. Due to imperfections in cataloging and metadata, detecting such overlap reliably is a very hard problem [1]. Dealing with multiple sources of metadata for better coverage only compounds this problem. Common issues with metadata include inconsistent formating of author and title field, incorrect data, confusion around multi-volume books and multi-book volumes, typos, etc. We found that the only truly reliable way to establish relationships between two books is to compare them side by side. Unfortunately due to the size of modern digital corpora, it is inconceivable to be able to compare every pair of books, and even less every pair of pages. To illustrate this point with a quick back-of-the-envelope calculation, imagine if we ran all the pairwise comparisons on the Google Books corpus. That corpus contains more than 15 million books, with an average of 330 pages per book. We could be comparing more than $5 \times 10^9$ billion pages. Direct $n^2/2$ comparison would yield $112.5 \times 10^{12}$ book pairs, and $12.5 \times 10^{18}$ page pairs. If we could afford 1ms per comparison, it would take 3567 years to compare all book pairs, and 777 million years for all page pairs.

In this paper we describe the method we use to cluster a corpus of more than 15M books, that relies on comparing similar books, both the book and the page level to determine book to book relations. This work has many practical applications. It can be used to improve metadata-based clustering by providing an orthogonal signal for the comparison of two books. Other applications include picking the best quality scan out of an equivalence class of similar books, catching anomalous books, detecting different but related books that share content, detecting piracy, etc.

## II. Prior work

The first part of the paper presents an efficient technique for finding similar books and pages over a large corpus. There has been much research on document matching. The general idea of using min-hashing has been successfully applied to many problems of media matching including audio search [2], and more recently image search [3], [4].

The second part of the paper focuses on automatically determining the kind of relationship between two matching books from the first phase: exact match, exact content match but different pagination, etc. Most prior work focused on detecting similar documents, often based on synthetically generated data, or documents with rather uniform properties (web pages, etc.).

Cross-document classification has been studied before [5], but it was studied as relation of texts and their semantics, rather than in books where pagination is much more important. We define a classification that is adequate for a large corpus of books with wide variations in input materials, format and quality.

## III. Book similarity detection

Our approach to make this problem tractable at large scale is based on two key building blocks: Min-hashing and

Mapreduce.

## A. Min-Hashing

Locally Sensitive Hashing (LSH) [6] is a method commonly used to perform approximate dimensionality reduction when dealing with vectors of high dimension. LSH is based on a family of hash functions $H$ where for each hash function $h \in H$ holds, and any two vectors $a, b$:

$$P(h(a) = h(b)) = sim(a, b).$$

LSH can be used for the calculation of set similarity. In our case, we have a set of extracted text features, which can be represented as a bitfield. The dimensionality is high: the number of all possible 5-word shingles is huge. Min-hashing [7] uses an LSH family of hash functions commonly used for set similarity estimates. Given set $A$, and $B$:

$$P(MinHash(A) = MinHash(B)) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid} \qquad (1)$$

which is also known as Jaccard similarity. To obtain the min-hash value given hash function $h$, we calculate min-hash on set $S$ as follows. First we calculate the hash values for all elements of $S$, and then we pick the minimum of those hash values. In other words $MinHash(S) = min(h(S))$, where $h(S)$ is the set of hash values of elements from $S$. Since the hash function is random, the probability that the hash value for any given element from $S$ is minimal is $\frac{1}{\mid S \mid}$. That means the probability that min-hash is in some $S' \subset S$ is $\mid S' \mid / \mid S \mid$. Since $MinHash(A) = MinHash(B)$ will match only if the element whose hash value was minimal is in $\mid A \cap B \mid$, the probability of min-hash collision will be $\mid A \cap B \mid / \mid A \cup B \mid$, which proves Eq. 1. By applying multiple independent hash functions, we can then find a good approximation to the similarity metric we want. So for $N$ hashes used and $C$ collisions detected we may estimate similarity of two sets to be $C/N$. Min-hashing allows us to reduce dimensionality and therefore reduce the amount of data we manipulate, which is crucial when dealing with 5B pages (15M books x 330 pages).

## B. Features

Our similarity detection scheme is sensitive to both the number of min-hashes and, the choice of features that we use to represent the document. Picking features that are too common between unrelated documents results in many false positives, and results a quadratic expansion of collisions, that may degrade performance to the point where it becomes intractable. For example, we could decide to use the words themselves as features, which would be a bad idea because many words are common across any two books of the same language. At the other extreme, picking overly discriminatory features increases the sensitivity to OCR mistakes, imprecision in layout analysis, and other types of errors that are common with document digitization.

So far, we have mainly experimented with text features, which work well for roman text. Each feature extractor operates on normalized text, derived from the raw OCR. For example the normalized form of '(Nice) Day !' would be 'nice day'. By stripping the non-alphanumeric characters and lowercasing the entire document we increase the likelihood of matching documents with little loss of precision. We have had our best results so far with a family of features based on a sliding window of n-grams for both words and characters. The words were represented as consecutive runs of characters separated by word-delimiters. In Figure 1 we show degradation of the min-hash based similarity measure for identical pages with an artificially introduced character error ratio (CER). We do the measurement for both word n-grams (referred here as *word shingles*), and character n-grams, on four common scripts in the corpus. One can see that the similarity measure quickly degrades with an increase of both CER and text length of an average feature. Word shingle degradation depends on average word length, which varies from script to script; character n-grams are invariant. However we prefer word shingles over character n-grams because word shingles produce 6-12 times fewer features than character n-grams, resulting in faster min-hash calculation.

The optimal choice seems to be to use word shingles of size 5. Each shingle was represented by its 64bit fingrprint to speed up calculation. Word shingle feature has been successfully used before on a similar corpus [8], which was one of main driving factor for us to use it.

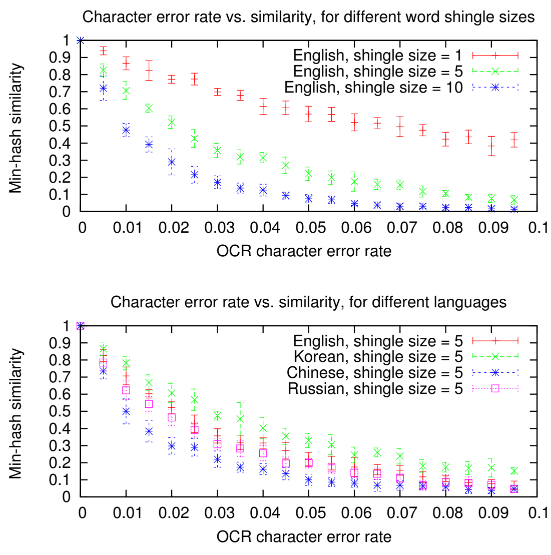
## C. MapReduce

MapReduce [9] is a software framework for running computation on large data sets on clusters of machines, allowing us to process data in parallel utilizing hundreds of CPUs. MapReduce schematically runs in two stages: a Map phase which transforms a set of key-value pairs into another set of output key-value pairs and a Reduce phase in which all the values with same output key are merged. The framwork handles splitting the work, scheduling it, and moving data around to make it efficient. It is suitable for many real world problems, including the one discussed in this paper. MapReduce has many open source implementations including Hadoop and Twister.

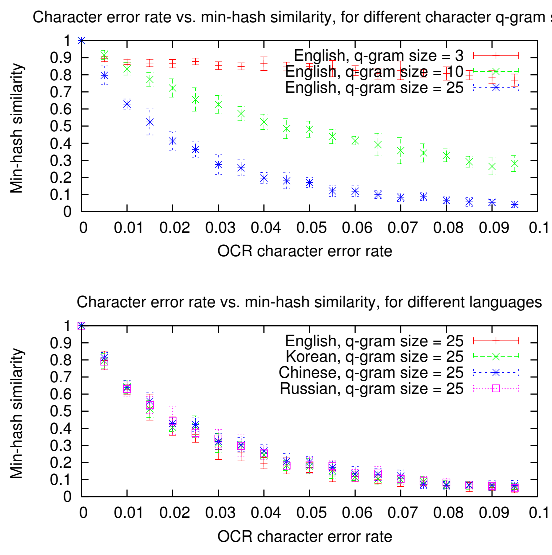
## D. Algorithm Overview and Process Flow

Our algorithm for detecting similar documents (either books or pages) is shown in Algorithm 1. There are two stages. In the first we extract document features and calculate a fixed number of min-hashes, and in the second we calculate all colliding pair similarities based on min-hash collisions.

The general process flow of the framework is shown in Figure 2. The first map-reduce extracts text-based features, and computes their min-hash values per-book (100 hashes) and per page (34 hashes). The second map-reduce derives



(a) Dependence of min-hash similarity on OCR error rate, word shingle size and language



(b) Dependence of min-hash similarity on OCR error rate, character n-gram size and language

Figure 1: Effects of of CER, language, and text features on min-hash similarity

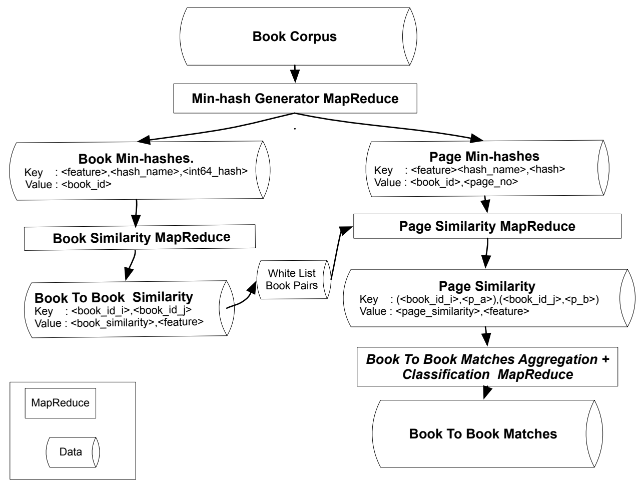

Book Corpus
Min-hash Generator MapReduce
Book Min-hashes.
Key : <feature>,<hash_name>,<int64_hash>
Value : <book_id>
Page Min-hashes
Key : <feature><hash_name>,<hash>
Value : <book_id>,<page_no>
Book Similarity MapReduce
Page Similarity MapReduce
Book To Book Similarity
Key : <book_id_i>,<book_id_j>
Value : <book_similarity>,<feature>
White List Book Pairs
Page Similarity
Key : (<book_id_i>,<p_a>),(<book_id_j>,<p_b>)
Value : <page_similarity>,<feature>
Book To Book Matches Aggregation + Classification MapReduce
Book To Book Matches
MapReduce
Data

Figure 2: Process flow overview

book similarities based on min-hash collisions. Note that we use the list of similar books from the first step as a filter, to quickly remove books with no overlap. The third map-reduce produces page similarities, and finally, the book and page similarity information gets accumulated for each book pair. From those we can derive the signals described below and perform classification.

## IV. Book To Book Relation Classification

### A. *Definitions*

Accurately describing book-to-book relationships can be a daunting task. Books are a long form of content that often lives through multiple editions, rewrites, etc. Two pairs of books may have an identical degree of similarity, but for one pair the differences are manifested as different pagination, whereas for the other they are due to the addition of a preface.

We are ultimately interested in two axes. One is the degree of pagination matching, because it lets us accumulate information over multiple copies of a page across the matching books. The other is the amount of content overlap, because high content similarity, regardless of pagination, gives insight into book matching as whole.

This leads us to classify books with some amount of similarity into four categories:

**SAME_PAGINATION** Same text, same pagination (e.g. same editions, possibly different re-prints with trivial changes).

**DIFFERENT_PAGINATION** Same text but paginated differently, typically to fit in a different form factor.

**CONTIGUOUS_SUBSET** One book contains the other with the same pagination (one could be volume 1, and the other volumes 1-3 of a multivolume work)

**OVERLAPPING_TEXT** The two books have a large amount of overlap, many pages in a row for instance, but they do not fit in the categories above (e.g. same story in two different collection of stories).

**Algorithm 1** Simplified similarity detection algorithm

```
1: $kCount$                    ▷ Number of min-hash functions used
2: $hash\_map \leftarrow \emptyset$          ▷ Map of hashes to vector of id's
3: $count\_map \leftarrow \emptyset$         ▷ Map of id pairs to number of colliding hash functions
4: $sim\_map \leftarrow \emptyset$       ▷ Map of id pairs to their similarity
5:
6: // Stage 1) Extract features / Calculate min-hashes /
7: //              Index by min-hashes
8: for all $doc \in book\_corpus$ do
9:     $features \leftarrow ExtractFeatures(doc)$
10:    $minHashes \leftarrow MinHashFeatures(features,$
11:                                   $kCount)$
12:    for all $hash\_key \in min\_hashes$ do
13:        $hashMap[hash\_key].push\_back(book.id)$
14:    end for
15: end for
16:
17: // Stage 2) Expand hash collisions
18: for all $hash\_key \in hash\_map$ do
19:    $ids \leftarrow hashMap[hash\_key]$
20:    for $i \leftarrow 0$ to $ids.size()$ do
21:        for $j \leftarrow i + 1$ to $ids.size()$ do
22:            $key \leftarrow ids[i] > ids[j]$ ?
23:                    $(ids[i], ids[j]) : (ids[j], ids[i])$
24:            $count\_map[key] \leftarrow count\_map[key] + 1$
25:        end for
26:    end for
27: end for
28:
29: // Stage 3) Collision counts to similarity
30: for all $id\_pair$ in $count\_map$ do
31:    $sim\_map[id\_pair] \leftarrow count\_map[key]/kCount$
32: end for
```

### B. Signal Extraction

To perform book to book relation classification we used a number of signals, some of which we will discuss here.

*1) Book Similarity and Page-based Book Similarity:* We define Book Similarity as the text similarity between two books. Ideally, books that share the same text should have Book Similarity close to 1 regardless of whether they share the same pagination.

Page-based Book Similarity is the average page similarity across matching pages for two given books. It provides some information about pagination. For instance if it is low while Book Similarity is high, it means that the pages are most likely going in and out of phase due to differences in pagination.

*2) Linear Fit:* We model book text uniformly distributed across pages with some offset from page zero. This way one can assume pagination of the similar pages to look like:

$$p_1 \in book_1, p_2 \in book_2, p_2 = a \times p_1 + b \tag{2}$$

Parameters $a$ (linear fit slope), and $b$ (offset) provide a useful signal. But so does the deviation in maximum predicted page:

$$\triangle page\_count = max(p_2) - (a \times max(p_1) + b) \tag{3}$$

Large deviation of $a$ from 1.0 is indicator of the different pagination, while large $b$ (offest) or $\triangle page\_count$ may indicate that $book_1$ is subset of $book_2$.

*3) Relative Consecutive Page Correlation:* 'Different pagination' books typically have partially overlapping content from page to page, in a pattern that wraps around periodically over the course of the book. In particular, pages of the book with higher density will often map to two pages in the other book. We define a metric, which we call *relative consecutive page correlation* to capture this. Similar paginations result in higher correlation. So if $p_i \in book_1$ (higher text density book), and $p_j \in book_2$, then $\exists(i,j), sim(p_i,p_j) \neq 0 \wedge sim(p_i,p_{j+1}) \neq 0$. If for all such $(i,j)$, we sum $sim(p_i,p_j)+sim(p_i,p_{j+1})$, and scale it with total number of pages in $book_1$ we get relative consecutive page correlation.

### C. Classification

As it will be demonstrated in section V-A, individual signals show good potential, but none of them is precise enough to be used by itself. We run a multi-class classification based on the all of the signals. For each class we calculate the confidence of having that class, and then we pick the class with highest confidence, or set it to 'overlapping text' if the maximum confidence is too low. Confidence is calculated as:

$$confidence_R = \prod_{i \in 0}^{K} \phi_{R,i}(s_i), \ \phi_{R,i} : \mathbb{R} \longmapsto \mathbb{R}_{[0,1]} \tag{4}$$

Where $\phi_{R,i}$ is manually specified for each signal used, and relation we defined. Most of the functions $\phi_{R,i}$ may be represented as low-pass, high-pass or band filters, with appropriate threshold parameters. For example for the mean page similarity signal (call it $s_{mps}$), and same pagination book relation (call it $R_{SP}$), we have $\phi_{R_{SP},mps}(s_{mps}) = max(0, 1 - (\frac{1-s_{mps}}{0.4})^2)$. This is a high pass filter, that removes everything below $s_{mps} = 0.6$. It may seem that we need to design many $\phi$ functions (filters) manually. In fact, only a few signals are relevant for each relation.

Note that there is a gray area here on how much overlapping text we require for two books to be marked as 'different pagination'. In a few cases, two editions may have over 20 percent distinct text in preface, notes, etc, which would cause us to mark two similar books as 'overlapping text'.

## V. Results

Our classification evaluation is based on 300 ground truth book pairs, uniformly sampled from set of similar books. Our training set consisted of around 1,600 ground truth pairs, and was chosen in less random fashion as we added pairs of interest to test the various features of our classifiers. In both sets relationships were ground-truthed and double-checked through manual side-by-side comparison. Each manual book pair comparison takes around 4min which is the limiting factor on the size of the ground-truth. The ground-truth is only about 0.01% of the entire corpus.

### A. Signal Evaluation

Various signal distributions for the ground truth training data are shown in Figure 3.

As we can see in Figure 3a, the Linear Fit slope for the 'same pagination' class remains very close to 1.0, as expected, while it spreads out much more for other classes. Similarly, in Figure 3b, we see that Relative Consecutive Page Correlation is greater than zero for the 'different pagination' class, while it's almost always zero for the 'same pagination' class. Finally in Figure 3c we distinctly see clusters. Book Similarity spreads out widely for all classes, mainly due to the frequency of OCR errors. But in combination with Page-based Book Similarity, it is easy to separate 'same pagination' books from other classes.

In Figure 4 we present precision/recall curves for individual signals. They behave pretty well, but for precisions over 90%, recall drops significantly. One exception is Page-based Book Similarity for the 'same pagination' class (Figure 4b), but even then precision drops significantly when the books are very similar. This particular effect is caused by cases where one of the books is a subset of another with some fraction of additional pages (similarity will be 1.0 even though the two books are different). We implemented early detection for such classes, which helped to remove noise from the final multi-class classifier.

### B. Clasification Evaluation

We initially tried multiple standard machine learning classifiers: Naive Bayes, various SVM schemas, and Winnow. None of them showed satisfactory precision/recall; this may be due to several factors, including the limited amount of available labeled (ground-truth) data, gross imbalances in the sizes of the various classes, and insufficient tuning of parameters. While it is likely that we could have made the machine-learning techniques to work well by appropriate measures (e.g., importance sampling, alternative choice of SVM kernel, etc.), we found that the simple alternative of devising a hand-tuned formula resulted in adequate performance, by combining the individual classifiers into a multi-class detector, factoring in some of the intuitive understanding that we developed after looking at many examples. Table I shows Precision/Recall numbers for the manual classifier, as well as for the Margin Support Vector Machine (one of the best performing machine learning schemes in our data set) classifier. The Margin SVM shows relatively good performance for the 'same pagination' class, but did not perform as well for the other classes, because we used a black-box implementation of the classifier and did not tune it thoroughly. The manual classifier was tuned for high precision on the classes that we intended to later make use of, but other operating points could be chosen. In the case of manual classification we explain the relatively lower recall in for the 'different pagination' class by the fact that different pagination tends to correlate with changes of edition, in which the addition of a large preface, or dozens of pages of appendix is not uncommon. These books can easily misclassified as 'overlapping text'. Another common source of confusion for our classifiers is those books that change in significant but subtle ways from edition to edition, such as reprints of yearly books, manuals, official documents with a different state name. We tend to classify them as 'same pagination' when they actually are different books and therefore 'overlapping text'.

Table I: Classification evaluation (300 ground truth matching pairs) for the manual classifier and *Margin Support Vector Machine* classifier (trained on 1,600 ground truth matching pairs).

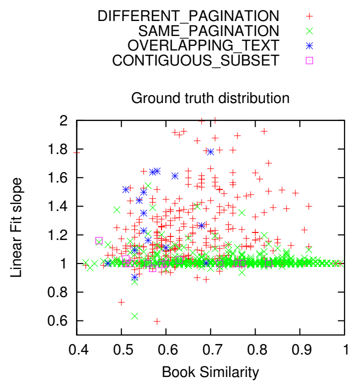
| | | Manual | | Margin SVM | |
|---|---|---|---|---|---|
| Relation | G.T.% | Prec. | Recall | Prec. | Recall |
| $SAME\_PAGINATION$ | 40.3 | 98.2 | 88.4 | 96.3 | 97.4 |
| $DIFFERENT\_PAGINATION$ | 16.3 | 92.3 | 73.5 | 82.6 | 87.5 |
| $CONTIGUOUS\_SUBSET$ | 7.7 | 95.2 | 86.9 | 44.4 | 26.7 |
| $OVERLAPPING\_TEXT$ | 36.0 | 78.6 | 96.3 | 100.0 | 6.2 |

### C. Full Corpus Run

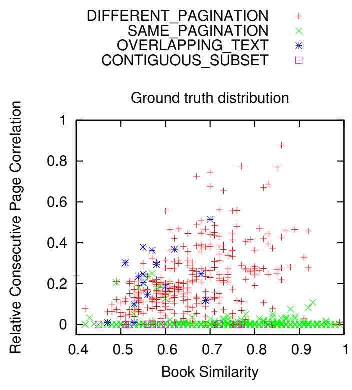
The relative size (scaled down by 15+M books that that we had at the run time) of the classes is shown in Table I in column $G.T.\%$. It is interesting to note the abundance of 'same pagination' books. It is expected since before digital typesetting, publishers would often just reuse the plates across reprints or editions. The proportion of clusters with 2 books, regardless of class is about 10% of all books. Small clusters of ten or smaller tend to be composed mostly of books with 'same pagination'. But as clusters get bigger, up to 40 or 50 books, they turn into mixed sets of books with different pagination. One consequence of this work is that we effectively detect clusters of identical books, with either same or different pagination. We can use both of those to enhance the 'master' copy by mixing and matching pages or paragraphs across copies.

## VI. Run Time Analysis

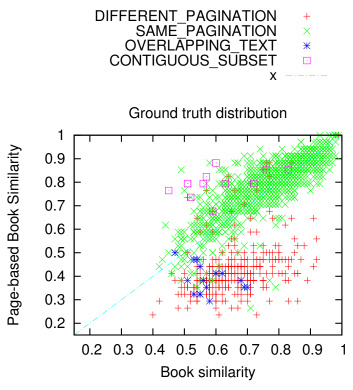
The similarity detection pipeline was all pair similarity search using the MapReduce [9] framework, on typical



(a) Book Similarity vs. Linear Fit slope

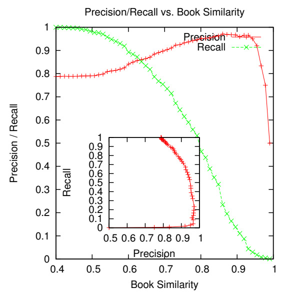


(b) Book Similarity vs. Relative Consecutive Page Correlation

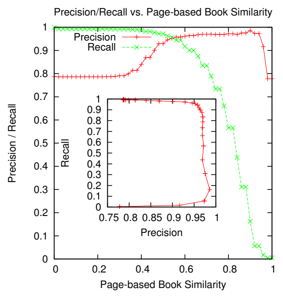


(c) Book Similarity vs. Page-based Book Similarity

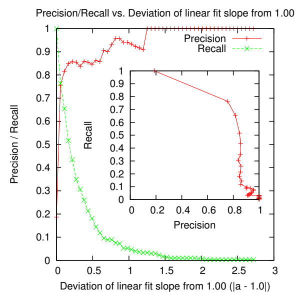
Figure 3: Ground truth (training set) distributions in signal space (each point represent a book pair)



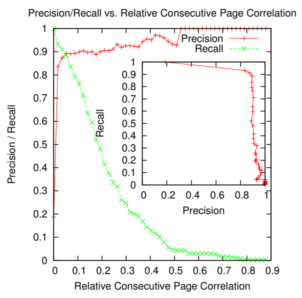
(a) Book Similarity for SAME_PAGINATION



(b) Page-based Book Similarity for SAME_PAGINATION



(c) $|\ 1.0 - a\ |$, deviation from 1.0 slope in Linear Fit for DIFFERENT_PAGINATION



(d) Relative Consecutive Page Correlation for DIFFERENT_PAGINATION

Figure 4: Precision/Recall for individual features (on training set)

google cluster (similar to those described in [10]). Measured in the terms of cpu-hours and our corpus of more than 15M books reading stored corpus takes 11275[1] cpu-hours; text feature + min-hash extraction takes 1428[1] cpu-hours (assuming that books are loaded already in RAM); the book similarity calculation takes take 72[1] cpu-hours; the page similarity calculation takes 11047[1] cpu-hours; the page based book similarity (including aggregation of page matches for each matched book pair) takes 1633[1] cpu-hours; and aggregation of matched book pairs (from each book to all matching books) and relation classification takes 406[1] cpu-hours. The final clustering of the books is done on a single CPU and it takes about 1 cpu-hour, with majority of time spent reading data from remote servers. Excluding the reading phase, we spent on average 3.5 cpu-seconds per book in the pipeline.

[1]This is the total run time of mapreduce multiplied by number of machines used. It does not exactly reflect the real per-CPU time because some of the workers complete faster than others, but provides a reasonable upper bound.

## VII. Further Work

An obvious goal is to make manual classiffier redundant and rely only on machine learning techniques. For this we need to gather more ground truth, or possibly use manual classifier results as noisy-ground truth on which we can do machine learning. In addition to tuning the machine learning, we could greatly improve classification by using features that are not necessarily text based. For example bounding box based signatures introduced by Spasojevic, Poncin and Bloomberg (2011) in [11] could be used in a complementary way with text-based features to improve page matching signalin situations where the CER of the given script is high but the word segmentation good, or on pages where it is hard to get text flow correct (e.g. title page, table of contents).

## VIII. Conclusion

In conclusion, we described a highly scalable mechanism based on min-hashing to extract features out of pages and books for the purpose of evaluating their similarity. We further described a variety of signals based on page/book similarities, which can be used to classify book pair relations

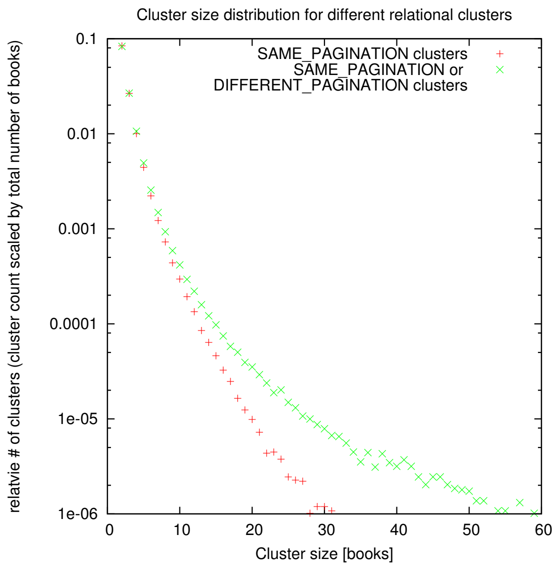


Figure 5: Cluster size distribution for different classes

between similar books, from very simple (Linear Fit), to reasonably sophisticated (Relative Consecutive Page Correlation). We showed that these signals achieve decent precision and recall when taken in isolation, but judicious combination helps boost detection results significantly against a manually classified set of books.

## IX. Acknowledgement

The authors would like to thank Dan Bloomberg who spent hours consulting with us and contributing his ideas to this project.

## References

[1] "Books of the world, stand up and be counted! all 129,864,880 of you." http://booksearch.blogspot.com/2010/08/books-of-world-stand-up-and-be-counted.html, August 2010.

[2] S. Baluja and M. Covel, "Content fingerprinting using wavelets," in *Proceedings of 3rd European Conference on Visual Media Production*, vol. 3, November 2006, pp. 198–207.

[3] A. Z. Ondrej Chum, James Philbin, "Near duplicate image detection: min-hash and tf-idf weighting," in *Proceedings of the British Machine Vision Conference.*, 2008.

[4] S. Baluja, M. Covell, and S. Ioffe, "Permutation grouping: intelligent hash function design for audio and image retrieval," in *ICASSP*, 2008, pp. 2137–2140.

[5] D. R. Radev, "A common theory of information fusion from multiple text sources step one: cross-document structure," in *Proceedings of the 1st SIGdial workshop on Discourse and dialogue*. Morristown, NJ, USA: Association for Computational Linguistics, 2000, pp. 74–83.

[6] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1998, pp. 604–613.

[7] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *Journal of Computer and System Sciences*, vol. 60, pp. 327–336, 1998.

[8] O. Kolak and B. Schilit, "Generating links by mining quotations," in *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, June 2008.

[9] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[10] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *Micro, IEEE*, vol. 23, no. 2, pp. 22–28, 2003.

[11] N. Spasojevic, G. Poncin, and D. Bloomberg, "Discrete point based signatures and applications to document matching," in *ICIAP 2011: Proceedings of the 16th international conference on image analysis and processing*, 2011.