Generating Links by Mining Quotations

Okan Kolak and Bill N. Schilit
Google Research
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA
{okan,schilit}@google.com

ABSTRACT

Scanning books, magazines, and newspapers has become a widespread activity because people believe that much of the worlds information still resides off-line. In general after works are scanned they are indexed for search and processed to add links. This paper describes a new approach to automatically add links by mining popularly quoted passages. Our technique connects elements that are semantically rich, so strong relations are made. Moreover, link targets point within a work, facilitating navigation. This paper makes three contributions. We describe a scalable algorithm for mining repeated word sequences from extremely large text corpora. Second, we present techniques that filter and rank the repeated sequences for quotations. Third, we present a new user interface for navigating across and within works in the collection using quotation links. Our system has been run on a digital library of over 1 million books and has been used by thousands of people.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*Architectures*; H.4.3 [**Information Systems Applications**]: Communications Applications—*Information browsers*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Miscellaneous*

General Terms

Algorithms, Design.

Keywords

automatic hypertext; link generation; quotations; digital libraries; hypertext.

1. INTRODUCTION

The world's libraries hold an estimated 32 million unique books, along with many millions more newspapers, magazines, and pamphlets [13]. This enormous cache of information represents 500 years of printing that preceded the digital era. Recently libraries, schools, corporations, and other organizations have started to scan this text and make it available online. Over 3.4 million pages of the New York Times, from the first issue in 1851 to current are now on the Web. Millions of books have also been scanned, digitized and made available.

This new material is creating an increasingly large text-rich but hypertext poor region of the web. When books are scanned they are indexed for search and generally go through automatic link generation to support navigation. However, manufactured links are often less preferred, in terms of quality, than their man-made equivalent [10].

It is possible to bridge automatic and manual link generation by mining existing links from documents. For example, table of contents, back of the book indices, and citations are generally high quality relations because they were manually created by authors and editors. Mining citations provide high quality cross book links [12], unfortunately, many books do not have citations. Moreover, citations are not evenly represented in a corpus, they tend to be more prevalent in the sciences than in the humanities.

This paper presents a new method for automatic hypertext based on mining quotation. Our approach shares many advantages of citation indexing. Quotations, like citations, were selected by authors so they are quality relations between texts. Quotation indexing also has distinct advantages over citations. Linking quotations provide a hyperlink target within a text, which is of particular benefit when the average length of a book is many hundreds of pages. Moreover, our analysis shows that quotations have wider coverage in a library of general books.

We divided the problem of mining and linking quotations into three sub-problems. First is mining candidate quotations from millions of books in a scalable and efficient manner. Second, is the problem of filtering non-quotations and ranking the remaining by quality. Third, is the problem of exposing users to the link structure in a clear and effective manner.

In this paper we cover each of these problems in turn. In Section 2 we describe large scale mining of repeated text sequences. Section 3 covers filtering and ranking of the candidates, and Section 4 covers the user interface. Sections 5 and 6 provide evaluation and related work. In Section 7 we talk about future directions such as incremental processing, ranking, and primary source identification.

HT'08, June 19-21, 2008, Pittsburgh, Pennsylvania, USA.

The system we developed has been run on over 1 million books that are part of Google's Book Search product. Because this digital library includes other automatic hypertext links we were able to compare usage, which we believe reflects user preference. We found that quotation links are one of the most popular link types.

We believe this is the first large scale system that mines and links quotations, the largest quotation collection ever created, and the first comparison of coverage and usage across generated link types in a real world corpus.

2. MINING REPEATED SEQUENCES

While it is possible to identify some quotations using surface text features, such as quotation marks, block indentations, font changes, etc. this approach has several drawbacks. First, not all quotations are clearly marked. Also, quotation styles vary from language to language and across authors, moreover, the same punctuation is used to mark dialogue. Such techniques will also miss the quotation in the primary source. Finally, even if quotes could be identified by their appearance, this does not help linking all occurrences of quotations together.

For large multi-language collections we have found "text that repeats" is the most basic and reliable way to identify all quotations, although this will include some non-quotations. Therefore, instead of selecting based on surface text features, we first extract all sequences of words, above a certain length, that repeat in two or more books. We then take these candidates, filter out non-quotations, and rank the rest using many features including text properties.

Finding sequences that are shared by multiple books is essentially an n^2 problem that requires comparing the text of every book to every other book. It is worth noting that the literature on duplicate and similarity detection deals with corpora on the order of hundreds of thousands of documents, typically with a document size of one to tens of pages. Our collections might grow to tens of millions of books, which translates to hundreds of millions of pages. Therefore, it is crucial to find a scalable processing method. In this section, we present our extraction method that makes processing a corpus of this size feasible.

Mining repeated sequences of words has multiple phases. We first shingle[1] all the books in the collection, creating a shingle table. The table's rows are indexed by the shingle fingerprint, and the columns store all the locations in books where the shingle occurs. The next phase is to take each book and use the shingle table to find all other books where variable length runs of shingles, i.e., repeated sequences, occur. The third phase is to group nearby repeated sequences that satisfy certain criteria. The goal of this step is to group repeated sequences that are part of the same conceptual unit together.

2.1 Shingle Table Generation

The first phase of our algorithm is the generation of a shingle table. Each book is fed through a shingler, which parses the book, normalizes the text, and generates a stream of overlapping shingles where each consecutive shingle is shifted by one token with respect to the previous token. For instance, 2-shingles for the text "as an example" would be "as an" and "an example". Possible normalizations are lowercasing, removing punctuation and accents, stemming, removing stopwords, collapsing numbers to a single token, and so on.

Shingles are inserted into a big multi-map (shingle table) where the key is the fingerprint of the normalized shingle text. In the rows we store a minimal amount of information, $\langle B, i\rangle$, where B is the ID of a book where the shingle appears and i is the index of the shingle in that book. We call each row a shingle bucket. The $\langle B, i\rangle$ values in a bucket are all identical in terms of their normalized text, and provide the locations in all books where the same shingle (k-words) occur.

Creating this shingle table requires a linear pass through the corpus and a massive sort. A small efficiency is achieved by discarding shingle buckets with 1 entry (shingles that appear in only a single book) because they cannot contribute to any repeated sequence.

Processing shingles provides an efficient way to reduce comparisons in later phases. Once we define the shortest

[1]A k-shingle is k consecutive tokens from a document where consecutive shingles overlap like the "shingles" on a roof.

```
   Data: Document d, Global shingle table T
   Result: Shared sequences for d
 1 activeSequences ← ∅;
 2 shingles ← ShingleDocument(d);
   // Process shinles in document order.
 3 for 0 ≤ i < shingles.size do
 4 |   srcShingle ← shingles[i];
 5 |   bucket ← GetBucketForShingle(srcShingle, T);
 6 |   foreach seq ∈ activeSequences do
 7 |   |   nextPos ← seq.target.position + seq.length;
 8 |   |   nextShingle ← ⟨seq.target.docId, nextPos⟩;
 9 |   |   if nextShingle ∈ bucket then
10 |   |   |   seq.length ← seq.length + 1;
11 |   |   |   bucket ← bucket - nextShingle;
12 |   |   else
13 |   |   |   sequences ← sequences ∪ {seq};
14 |   |   |   activeSequences ← activeSequences - seq;
15 |   |   end
16 |   end
   |   // Any shingles left starts a new sequence.
17 |   foreach trgShingle ∈ bucket do
   |   |   // Ignore shingles from the source book.
18 |   |   if trgShingle ∉ shingles then
19 |   |   |   newSequence ← new Sequence;
20 |   |   |   newSequence.position ← srcShingle.position;
21 |   |   |   newSequence.target.docId ← trgShingle.docId;
22 |   |   |   newSequence.target.position ← trgShingle.position;
23 |   |   |   activeSequences ← activeSequences ∪ {newSequence};
24 |   |   end
25 |   end
26 end
   // Close any remaining active sequences.
27 foreach seq ∈ activeSequences do
28 |   sequences ← sequences ∪ {seq};
29 |   activeSequences ← activeSequences - seq;
30 end
31 return sequences;
```

Figure 1: Repeated sequence extraction algorithm for processing a single book. A slightly modified version of this algorithm allows processing the whole corpus using the MapReduce paradigm.

quotations that we are interested in mining, we can set our shingles to that size and eliminate all pairs of books that do not share at least one shingle. The trade-off is that large sized shingles are going to occur less frequently but cause us to miss short quotations or quotations with spelling variations and OCR errors. Since we are less interested in bon mots and more interested in longer quotations, we felt comfortable with a relatively large value. In our current system we choose a shingle size of 8 based on our observations that most repeated sequences smaller than 8 words are common and colloquial phrases, and not significant quotations.

2.2 Repeated Sequence Generation

The second phase of our algorithm extracts runs of shingles (sequences) that are shared between books. The algorithm processes a book at a time using the shingle table. We refer to the book that we are currently processing as the "source book" and all the other books as "target books." A repeated sequence is formed of one or more contiguous shingles that appear in the source book and at least one target book. We will describe the algorithm for processing a single book, and then explain how it is extended to the whole collection.

Logically, we process a single book by first generating a list of shingles in the order they appear in the book. We then take each shingle, in book order, and using the shingle table find all the other books where the shingle appears. Each $\langle B, i \rangle$ in a bucket will either start a new or continue an ongoing repeated sequence between the source and a target book. This algorithm is presented in Figure 1.

In practice, we process all books simultaneously in a massively distributed manner using MapReduce [7]. We start with the shingle table as input. For a given book, we will need the shingle buckets for all shingles in the book (this is the equivalent of the lookups from above). We achieve this by outputting each shingle bucket once for every book that appears in the bucket. In the reduce phase we will get one reduce call for each source book, and the values will be all the shingle buckets where a shingle from the book appears. These buckets are sorted on the positions of the shingles from the source book. At this point, we can use the algorithm we used for processing a single book with some minor modifications. We do not need the shingling step at line 1, because we start with shingle buckets. Lines 3 through 5 become a single `ForEach` line as we get the buckets of the source book directly and in order. However, since we discard unique shingles, we will not get all the shingles for the book. We call these gaps in the source shingle sequence "source gaps". Whenever we see a source gap in the shingle sequence we are processing, that means text in that region does not appear in any other book and we close all the active sequences, the same way we do in lines 27-30.

When we are processing the whole collection, shingles that are shared by a large number of books enormously increase both the intermediate storage and the number of books that needs to be compared. Quite often these shingles are common phrases that do not really indicate a real connection so we would like to drop them for efficiency.

However, a common phrase that appears in a repeated sequence should not cause that sequence to be broken into pieces. To address this, we drop the bucket contents for shingles that are too common and distribute "free passes" for those shingles instead. If we get a mismatch during sequencing, line 8 of the algorithm, because of such a shingle, we assume that the target book actually has a match but the matching shingle in the bucket has been dropped. Free passes are used only within sequences, not at the boundaries, to reduce the likelihood of false matches. For non-boundary shingles, the nearby matches provide additional evidence that the intervening text is a probable match. One drawback of dropping shingles that are too common is that any repeated sequence that is popular enough to push all its shingle above the threshold will be missed by the system.

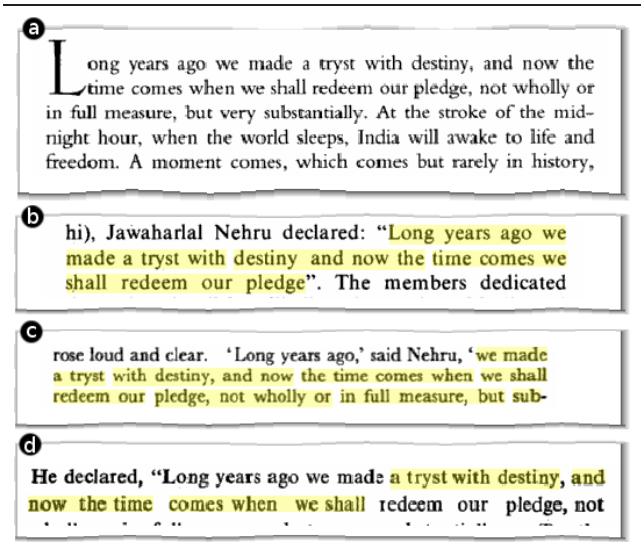

Figure 2: Examples of variations in quotation text across instances. (a) Reference instance; (b) Partial quotation; (c) Interjection; (d) OCR errors.

2.3 Sequence Grouping

The repeated sequences generated in Section 2.2 are based on exact matches across books. However, quotations tend to exhibit variations. For example Figure 2-a shows the beginning of the independence speech delivered by Jawaharlal Nehru, and 2-b through 2-d shows some variations we extract. Often times authors use different length segments from the original, quoting a single sentence, or even part of a sentence, as in Figure 2-b. Another issue occurs when an author splits a quote by introducing a phrase as in Figure 2-c. In other cases, variations in wording, spelling, or OCR errors cause an exact match to fail. Figure 2-d illustrates a case where some instances of letter 'e' are misrecognized, causing the sequence matching to terminate at these tokens.

As a result of these variations, when the sequence generation is complete, we usually end up with multiple distinct repeated sequences that actually come from a single quotation. For the example in Figure 2, we have three different repeated sequences. In order to group repeated sequences that are actually part of the same passage, we perform a sequence grouping phase. The goal of this operation is to bring the fragmented sequences together and identify the underlying complete passage, which may or may not appear in any of the individual fragments. For example, in Figure 2, sequences b, c, and d cover more together than they each cover individually.

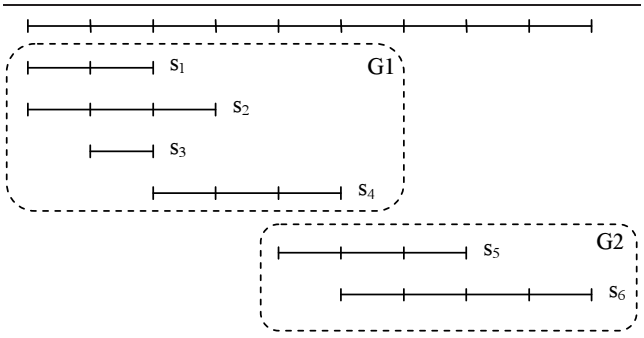

Figure 3: Sequence grouping process. Each segment of a line represents a shingle. The top line is the reference sequence in the source book. Other lines represent repeated sequences from different target books, in alignment with the same text in the source book. They are placed into two groups based on overlap in the source book. The two groups would merge if transitive grouping was allowed.

Let us illustrate the grouping process using Figure 3. For a given book, we examine repeated sequences ordered by their start position in the book. We create the first sequence group, G1, place the first sequence, s_1, in that group and make it the active sequence. We then start scanning through the rest of the sequences, adding sequences to the active group as long as they have a certain amount of overlap with the active sequence. Whenever we add a sequence that starts at the same location, but is longer than the active sequence, the longer sequence becomes the active sequence. For our example, s_2 is such a sequence and replaces s_1 as the active sequence. As we go down, s_3 and s_4 also become part of the group G1. When we reach a sequence, e.g. s_5, that does not have enough overlap with the current active sequence, we start a new sequence group, G2, make the current sequence, s_5, the active sequence, and continue the process.

One parameter of the grouping algorithm is whether we allow transitive grouping. In our example, overlap between s_4 and s_5 does not cause the two groups to merge, because s_2 is the active sequence of G1. If we allow transitive grouping, then s_5 would be connected to s_2 transitively through s_4 and would be included in the same group. We found out that transitive grouping tend to cause over-grouping, so we do not use it.

For each sequence group, we use the text spanned by all the sequences in the group as the text for the group. For G1 in Figure 3, the group text would be from the beginning of s_1 to the end of s_4.

3. FILTERING AND RANKING

At the end of the sequence extraction and merging phases we have created a huge database of sequences shared between two or more books. Many of these candidate quotations are copyright sentences, legal boilerplate, publisher addresses, bibliographic citations, and other text that repeats. These sequences are not interesting as quotations and should be filtered.

We have identified classes of such repeated sequences. Common cases include citations, publisher addresses, titles of other books by the author or publisher. These sequences usually occur in the front-matter and back-matter of books and not in the body, so one filter we apply checks whether the number of instances of a quotation that appear on such "low content" pages is above a certain threshold. Other common cases are tabular data, figures, formulae, and so on. We identify these using a basic form of language modeling: If the quotation text has some unusual characteristics, such as too many digits or special characters, repeated tokens, etc. it is eliminated.

We also perform filtering based on book editions, but this filter eliminates instances rather than complete sequences. All the instances of a repeated sequence that appear in other editions of the source book are filtered but the sequence itself remains unless all its instances are gone.

Once filtering is complete, we are left with a set of repeated sequences that look more like quotations. However, some are unexciting such as "The unemployment rate is the percentage of the labor force that is unemployed." while others are more interesting like: "All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood." Distinguishing these is difficult because it is hard to define what makes a good quotation so our approach is to rank the entire set, and try to give "good" quotations a higher score.

We started with basic intuitions about the length and frequency of quotations. If a sequence is too short, it is probably a common phrase or colloquialism. If it is too long, then it is probably a document of some sort that is included in books as a whole.[2] Neither makes an interesting, concise quotation. If a sequence appears in few books, it is probably not very important. If it appears in many books, on the other hand, it is again possible that it is a common phrase. We designed a simple scoring method based on these intuitions. We start by computing two scores, one based on sequence length and the other based on usage frequency. The overall score of the sequence is computed as a weighted geometric mean of the length and frequency scores.

For computing the length score of a sequence, we use a piecewise function that covers different ranges of the length space. First, we have sequences that are "too short." Any sequence in this range would get a low score. Next we have the range of "desirable lengths," where the score increases with length, reaches some peak value, and starts decreasing again. After that we have a "too long" range, where scores are low again. The shape of the piecewise function is determined by experimental tuning. The frequency score is computed in exactly the same way. The only difference is the shape of the curve. Although very simple, this scoring method performs remarkably well as discussed in Section 5.1.

4. NAVIGATION USER INTERFACE

This research grew out of our work and experience with Google Book Search[3], one of the largest digitized libraries in existence. We saw the lack of authored links made it difficult for users to have a quality hypertext experience. We felt that navigating quotations, reminiscent of Ted Nelson's transclusions, might address these problems and provide an engaging form of hyperlink between books.

[2]Examples are speeches, short stories, historical letters, etc.
[3]http://books.google.com/

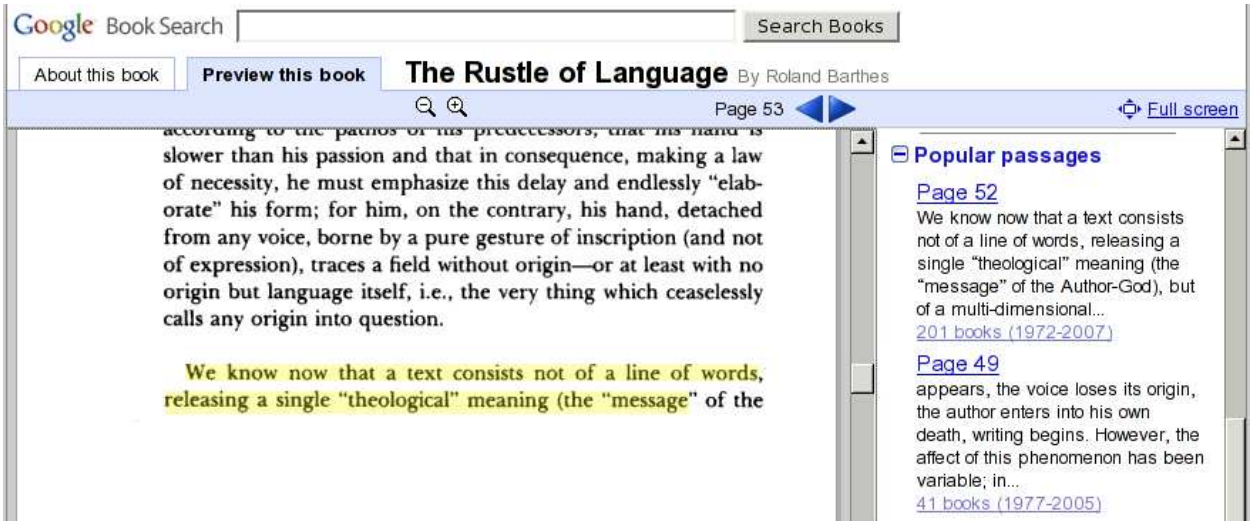

Figure 4: Popular Passages in the context of a book, allowing the user to see an overview of passages that appear in this book and quickly jump to their locations to read them in context. The quoted text is highlighted.

As we started the research our thesis was that if authors found the same words interesting enough to copy into their books then the books as a group probably share something interesting as well. By exposing these implicit relations we could replace the missing hyperlink network in the collection and allow users to follow a trail from book to book discussing the same words. We saw that these links are richer than the typical hyperlinks on the web since they themselves contain concise bits of information. Furthermore, they land the user in a context where the information in the link is very relevant. In our ideal, quotation based links would allow the user to explore various opinions and interpretations of the quotation in context across many authors.

As we designed the user interaction the first question we asked was how do we present this idea to users. The word "quotations" is descriptive but also misleading, since people think of Bartlett's, and ours is a much broader set. We arrived at the less loaded term of "popular passages" which gets across the idea that these are bits of the book that appear in other books.

As part of our design process we developed user interface mock-ups and showed them to users. One issue that arose was that a good display form for the extracted quotations was needed. Through the mock-ups we realized that within-book navigation might be as engaging as across-book navigation. We also saw that our design should allow users to jump to books where the quotation appears in a fluid way.

In this section, we describe the "Popular Passages" feature that launched on Google Book Search in September 2007. We also developed a UI for exploring ideas that emerge from quotations, which is described elsewhere [17].

4.1 Display Form of Quotations

The quotations that we mine do not generally look good. Frequently spurious words will appear at the front of a quotation because more than one author introduced it in the same way. For example, "Lincoln wrote," might be tacked onto the front of Lincoln quotes. Similarly, words may be incorrectly appended to a quote, and because of OCR errors and spelling variations, words might also be dropped. We call this problem of distinguishing between quotation text and context the "boundary problem."

One consideration when calculating the display form for a quotation is it's length. The extraction algorithm produces the longest run of shingles that are shared in two or more books. So this means that a quotation group might have a 6-sentence quotation along with a 2-sentence sub-quotation that is far more popular. It is desirable to display the shorter and more familiar words.

We address both of these problems statistically by computing two types of skylines. The first skyline has a block height based on the popularity of each shingle S in the longest instance of a quotation. We create a skyline vector that stores the count of each shingle S_i in all the instances. This tells how popular each piece of the quotation is, and will, for example, show that a sentence at the start of a long quotation is the most quoted form.

The shingle skyline is also used to address the boundary problem. Generally if the longest version of the quotation has additional words that are part of the context ("Lincoln wrote"), they will not appear in most of the instances. So the technique to compute the most popular form also removes spurious context.

In addition to computing a skyline based on shingle frequencies we also compute one based on punctuation frequencies. The punctuation skyline counts the popularity of quotation marks at the various shingle positions. We found that these are also good places to start or stop a quotation when figuring the display form.

4.2 Navigation within Books

Users of digitized off-line document collections, especially the collections of digitized books, face a problem that is usually not an issue for on-line document collections: Navi-

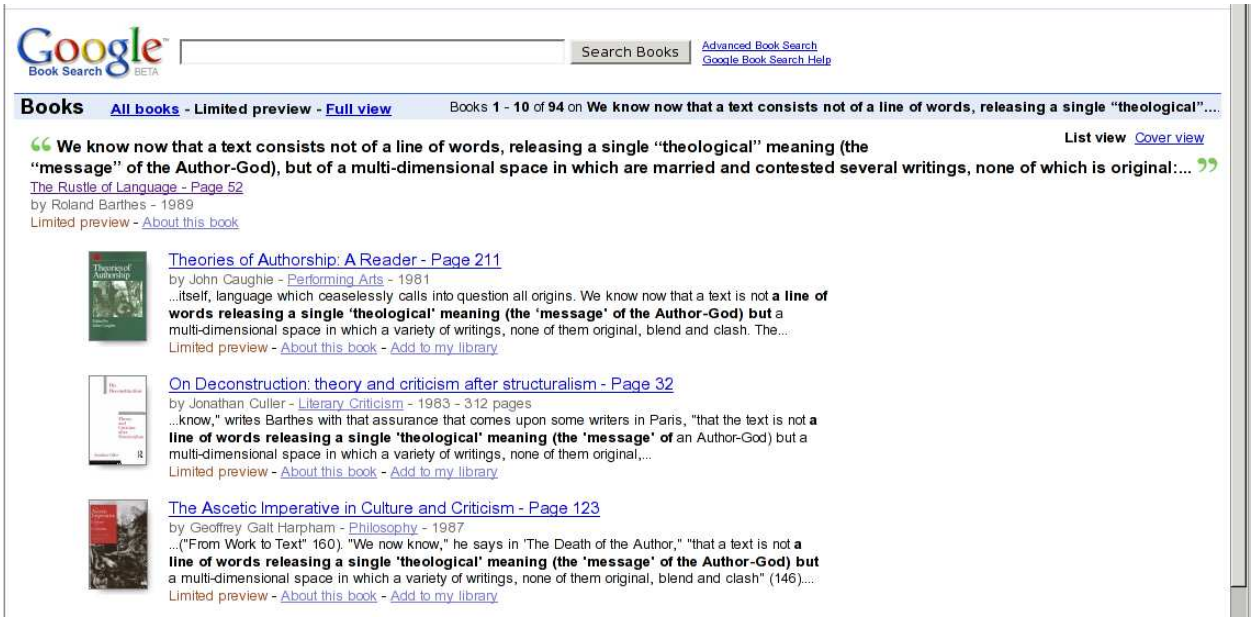

Figure 5: Interface for navigating between books using Popular Passages as a pivot point. The user lands on this page via popularity links in Figure 4, and is presented with the list of books where the quotation appears. Clicking a title takes the user back to interface in Figure 4, displaying the quotation in the context of the selected book.

gating within documents. For collections of books, landing on a document is often times not enough. When faced with a book that has hundreds of pages, a user benefits from tools that can aid in (1) getting a general feel for the relevance of the book and (2) quickly navigating to the relevant/important bits within the book.

We believe Popular Passages provides one way to achieve both goals. A passage that is repeated in multiple books can be viewed as an important bit of information that has been selected by numerous authors. For the source book, this passage has been deemed more important than other passages in the book. For the secondary books, quotations are often used to support a claim, and idea, or a point the author is trying to get across, and therefore would be located at significant points in the flow of the text.

When the user is viewing a book we provide the top ranked popular passages that appear in the book as illustrated in Figure 4. We currently display the top three quotations as a collapsible module on the right sidebar, and provide a link to expand the list to top 10 quotations. Along with each passage we provide a "context link" that display the page number where it appears. When the user clicks on the page number, the view on the left jumps to that page, and the passage is highlighted. For each passage we also provide a way to navigate between books as described next.

4.3 Navigation between Books

Navigation between books through quotations are possible using a "popularity link" that provides some brief information about the spread of the quotation, such as the number of book it appears in, and the publication date range for those books (e.g. "41 books (1977–2005)"). You can see examples of these links on the right side of Figure 4. When the user clicks on one of these links, we present a page that is similar to search result pages, as can be seen in Figure 5. The header shows the passage and the book from which the request for this page originated. Below the header is a list of books where this quotation appears. Each result presents the version of the quotation that appears in that book along with some context.

The user can jump to the passage within a particular book by clicking on its cover or the title. In essence, context links and popularity links take the users between UIs illustrated in Figures 4 and 5, allowing them to see the quotations that appear in a book, or books that a quotation appears in.

5. EVALUATION

In this section we present evaluations of different aspects of our system. First, we ran an experiment with human annotators to provide analysis on the quality of the extracted quotations. Second, we collected real-world usage statistics on the Google Book Search web site where Popular Passages is deployed. Finally, we measured the coverage of extracted link types (quotation, citation, and URL) in our corpus.

5.1 Manual Quality Assessments

Evaluating the quality of extracted quotations is a complicated task. The vast majority of repeated sequences are of low value to users and we rely on filtering and scoring to eliminate them. Given the large number of repeated sequences, we decided to evaluate the precision of the filtered and ranked set rather than overall recall. In this experiment we evaluate the quality of passages retained after filtering.

This evaluation approach introduces a precision bias because filtered or low-ranked sequences will not be evaluated.

To reduce the bias we increased the number of low-ranked passages retained. Once we computed scores for the extended set, we sampled 120 passages from the low end of the score spectrum and 120 from the high end. We then asked people to rate these passages on a Likert scale of 1 to 5, along with a 0 response for passages that they cannot rate.[4] Evaluators were told that "A quality quotation is usually a brief portion of a written work or speech that is historical, noteworthy, or otherwise interesting. It may be helpful to look at the quotation in context." Scores were labeled as "Excellent", "Good", "Neutral", "Bad", and "Awful." Each quotation was evaluated by up to three people.

We first eliminated the quotations that were not rated by at least two people, reducing the number of quotations used for evaluation to 195. We treated 1 and 2 as good, 3 as neutral, 4 and 5 as bad. We labeled a quotation good or bad if at least two annotators marked it so.

After these steps we had 108 quotations marked good, 58 marked bad, and 29 where annotators did not have agreement. Inter-annotator agreement was high, reflecting that the task was not very difficult. We saw 85% inter-annotator agreement if we considered neutral score as disagreement and 92% if we consider neutral score as agreement.

The experiment showed that our system did a very good job of ranking: 88% of the passages our system extracted and ranked as good quotations are positively rated by human evaluators. We also found that 38% of the passages we ranked low were ranked as good or neutral quotations (i.e. false negatives). We speculate that the false negative rate appears higher than it really is because low ranked quotes were filtered, reducing the size of the negative sample set.

5.2 Utility to Users

In our second evaluation we were interested in comparing navigational features on Google Book Search: users can navigate through quotations (Quotations); to other editions of the same book (Editions); to books in a computed cluster (Related); or to books that cite the current book (Cited By). We measure active users over a 30 day period, avoiding a novelty effect by using a period well past feature introduction. As can be seen from Figure 6, popular passages has the most active users of all navigational features. It is important to note this evaluation does not account for feature coverage, presentation differences (i.e., placement on the page), or other differences. The conclusion we draw is that users find utility in popular passages and interact with them, however, because of the factors above, we aren't able to make strong claims regarding relative utility of these methods in general outside of our current system.

Similarly for within-book navigation, we observed that popular passages are used quite often by users. We do not provide a comparison figure for within-book navigation because the features that allow this are very diverse and their direct comparison is less meaningful.

5.3 Coverage

In our third evaluation we were interested in measuring the percent of books that include various link types. Earlier in this paper we asserted that quotations are the most common link type appearing in a general set of books. Our corpus consisting of over a million scanned books from thousands of sources is indeed general. Many of the books use URLs as a means to link to relevant or supplemental information, however this is a recent development. Citations and references have been used for hundreds of years but they appear more often in technical and scientific books.

We analyzed our collection to see the percentage of books that include each of these link types, Figure 7 illustrates the results. URLs appear in only 12% of the books.[5] Citations, including informal mentions of other books in passing, appear in 38% of books while 20% of books are cited by other books. Quotations, on the other hand, appear in 58% of the books in the collection, after we apply filter and ranking.[6]

The linking features presented in Figures 6 and 7 are not the same, which warrants some explanation. In Figure 7 we are only comparing the coverage of inter-book links that are author generated. Editions is not a link between different books, but rather versions of the same book. Coverage of related books (and other generated links) is less meaningful because it depends on the parameters of the methods used. Quotations and incoming citations (Cited By), however, can be contrasted in the two figures. Higher usage of quotations is partially explained by its higher coverage. But even when normalized for coverage, quotations have higher usage.

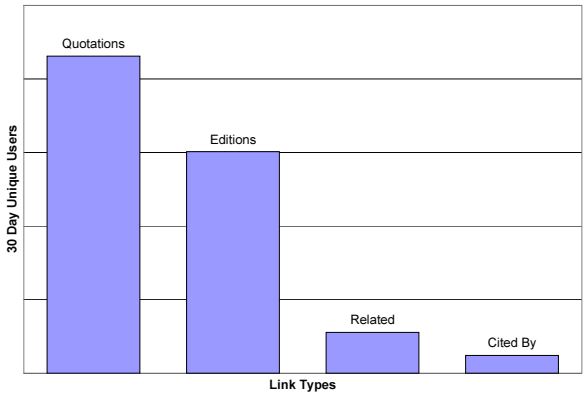

Figure 6: Relative 30 day unique user counts for various inter-book navigation features.

6. RELATED WORK

In this section we cover related work on automatically creating hyperlinks, followed by a discussion on methods for analyzing similarities between documents for copy and plagiarism detection.

6.1 Automatic Hypertext

Automatic hypertext generation has been an active research area with numerous articles, even journal issues [1]. Therefore we will provide a general discussion, and refer the reader to [2] and [23] for a good survey of the subject.

[4]The 0 category was used for quotations on which the evaluator was unable to make any judgement. For example, a quotation in a language the evaluator does not understand. If the evaluator understands the quote but cannot decide whether it is good or bad, the score would be 3.

[5]By URL, we mean HTTP and FTP addresses anywhere in the book. For example, even a book that only has the URL for the publisher at the back cover is counted.

[6]We note that not all of these quotations are of the same quality.

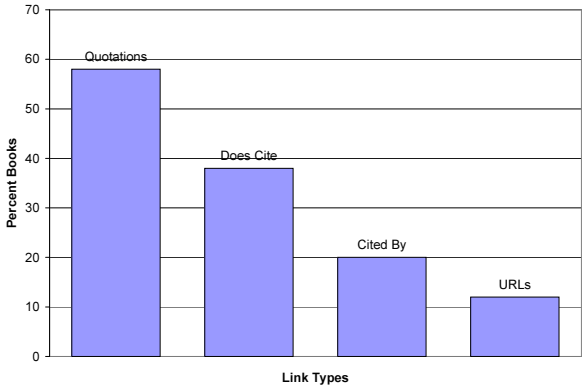

Figure 7: Coverage of various link types that appear in books.

There are three main categories of automatic link generation. The first category is extracting links and/or structure that already exists in the document. Here links are manually created by the author and the system simply identifies and turns them into clickable anchors. Most of these links are within the documents, with few exceptions such as citations that point to other documents. Some examples are table of contents and back of the book index; figures, tables, citations and their mentions in the document body; definitions of terms, formulas, etc. and their uses; citations and mentions of other documents.

The second category is where the document does not have an explicit link but links are created based on the structure and syntax of the document. Some examples include: identifying chapter headings and creating a table of contents that links to them; creating an index for words in a document pointing to where they occur; identifying definitions and linking occurrences of the defined object to definition; providing links to previous/next paragraph/section/chapter. This category is very close to the first one, the difference is that the author or editor did not create the link explicitly. Note that these links are still based on the existing structure of the document.

The last category of links are inferred without relying on explicit link or structure in the document. Some examples are: related documents using clustering; computing categories, subjects, or key terms from a documents and using this as a link; identifying semantic units of text from the document and linking them to other documents that are determined to be relevant. In this approach, the nature and the quality of the links are highly dependent on the way links are created, because there is no underlying human generated link in the document.

Popular Passages fall under the first category, or arguably the second, because we identify passages that are selected and quoted by the authors of the books. However, our approach is similar to inferred links in some respects. One goal of the inferred links is to link together relevant documents and concepts. Popular Passages also link related documents and related concepts in those documents together. However, Popular Passages are more powerful in this respect because they are based on manually authored semantic links between documents. On the other hand, they do not have the coverage of inferred links, which can be arbitrarily high depending on the relation extraction method used.

Popular Passages are also similar to Ted Nelson's idea of "transclusions" [16] in terms of benefits and usage. However, they are different in the way they are created. Transclusions come about by creating documents using an edit decision list structure, where inclusions from other documents are explicitly marked and linked at creation time. Popular Passages, on the other hand, are mined from existing text collections, allowing to link documents together regardless of how they are authored.

6.2 Similarity, Copy, Plagiarism Detection

Broder et al. introduce shingling to compute document resemblance and containment in an efficient way [4, 5]. They also present methods for using a subset of shingles while still allowing resemblance and containment computations. We use their basic shingling technique, however, our goal, processing, and intermediary data structures are quite different. We are interested in all the sequences of text two documents share, regardless of whether they are similar or not. Broder et al. compute the similarity of documents regardless of whether they share any significant sequence of text.

Suffix trees [11, 22, 14] allow very efficient substring matches and therefore can also be used to find repeated sequences. Unfortunately, distributed suffix trees are not practical, and a single suffix tree for large collections is not feasible because of size limitations. One alternative is to construct individual suffix trees for each document and compare the collection against it, as Monostori et al. does in [15]. While this approach solves the space issue, it is also not feasible due to execution time constraints.

Copy detection [6, 9, 3, 18, 20] and the closely related problem of plagiarism detection [8, 21, 19] have been studied by numerous researchers. While they appear similar to repeated sequence extraction at first glance, both involve identifying documents that are significantly similar.[7] Repeated sequence extraction, on the other hand, identifies all sequences shared by two documents regardless of whether the documents otherwise resemble each other. As a result, sampling and fingerprinting techniques used for copy and plagiarism detection makes them unusable for repeated passage extraction.

Fetterly et al.'s work on plagiarism detection [8] is closest to our approach, as it focuses on extracting phrase duplication. However, it uses sampling, does not retain all the content, and only identifies phrases of a given fixed length.

It is worth noting that none of the past work in this section does address the need of filtering and ranking interesting phrases. Furthermore, while copy detection is not suitable for repeated sequence extraction, it would be possible to use repeated sequence extraction to identify copies, whether they are full or partial.

7. FUTURE WORK

In this section we discuss some areas for future work.

7.1 Incremental Processing

Our current method is a batch process. For a dynamic collection where books are regularly added, the ability to

[7]While the problem of plagiarism detection would include sub-document level similarities as well, existing work generally focuses on document level similarity.

do incremental processing is desirable. One approach is to simply search for all the repeated sequences that are already identified in each new book. This requires the ability to search for partial matches efficiently due to variations in quote instances. But more significant problem is that newly emerging quotations will not be discovered.

A better solution is to add the shingles for the new books to the existing repository and run the sequencing algorithm for the new books using the augmented shingle repository. While this approach can identify some of the newly emerging quotations, it cannot identify all because of the fact that unique shingles are discarded. If a book introduces a new repeated sequence that had a single, non-repeated instance in the corpus before, shingles of the sequence that were unique will not be in the shingle repository. There is no way around this unless we store all the shingles, which increase storage significantly. For example, in our collection we found that a shingle size of 5 will produce 5 times more unique shingles than non-unique shingles. As the shingle size gets larger, the ratio of unique shingles get even larger.

Because of these issues, we are still investigating efficient algorithms for incremental processing.

7.2 Improved Ranking

Although our current ranking algorithm does a reasonable job, there is still room for improvement. Deciding whether a quotation is "good" or not is a subjective and difficult task. Once bad quotations are filtered out, simply looking at the length and the frequency of a quotation is often not enough. Other signals that help measure the actual information content of the quotations are needed.

Another aspect of the scoring that can be improved is the parameter estimation for the scoring function. Current scoring parameters were tuned manually because of insufficient training data to use machine learning algorithms. We would like to explore how to generate training data in a cost-effective manner and/or applying active learning techniques that can work with limited amounts of labelled training data.

7.3 Primary Source Identification

The extraction method presented here does not distinguish between primary and secondary sources. For a given quotation, the ability for users to read it in its original context, and finding other sources that analyze and discuss the original idea would be useful. Without primary source identification, the user has to manually sift through the books where a quotation appears.

When we display the list of books on which a quotation appears (Figure 5), we would like to show the source book where the quotation originated.

Primary source information can be useful for ranking, too. For a given book, the quotations that originated in the book are arguably more relevant and valuable than the passages that are quoted from other books.

If original publication date data was available for all the books in the collection, we might pick the book with the earliest publication date as the source of a quotation. However, some books have meta-data errors, some primary sources are republished, and some primary sources are not in the collection. Our system currently orders books by publication date, but we would like to introduce more reliable algorithms for primary source identification.

7.4 Attribution

An issue that is closely related to primary source identification is attribution. As users look through quotations, they would probably be interested in the authors of the passages. Proper attribution is also an important issue from a scholarly point of view. Primary source identification would aid in attribution tremendously, as the author of the primary source is often the author of the quotations that originate from that source.

Keyterm extraction from the text surrounding a quote described in [17] is a valuable tool for attribution. The author of a quotation is often among the keyterms, as many instances mention the author in close proximity to the quotation. Primary source and keyterms together provide a strong starting point for attribution. It remains to be seen how they would need to be combined and augmented to reliably attribute quotations.

8. CONCLUSIONS

Scanning books, magazines, and newspapers is a common activity that is producing huge amounts of digitized content that lack the hypertext links available in collections authored online. Our work with digitized book content made us look beyond current automatic hypertext techniques because we wanted to provide broader and more engaging discovery, navigation, and summarization experiences. We developed a new kind of hypertext link between quotations in books, which we launched as Popular Passages in Google Book Search.

In order to mine and link quotations we developed a method for extracting their fundamental positive trait, repeated sequences, from a book collection. These repeated sequences form the candidates for quotations. The extraction method does not rely on syntactic cues, book structure, or language specific processing; it is completely language independent, scalable and capable of processing over a million books in all languages. To the best of our knowledge, this is the first system capable of extracting repeated text segments in such a big and diverse text collection.

We also developed a ranking and filtering method that allows us to sift through millions of repeated sequences, which includes a wide range of textual repeats, and identify the quotations that are concise, engaging, and useful from end user perspective. Our evaluations show that 88% of the repeated sequences that we identify as good quotations are confirmed by human evaluators.

In this paper we have argued that quotations provide strong links between books and can be an engaging way for users to navigate between related books to see an idea or concept in different contexts. Quotations also support in-book navigation by allowing users to quickly locate the interesting bits. We designed an interface around these ideas to allow navigation within and between books in our collection using quotations as pivot points. The quotation based navigation feature is currently live on Google Book Search and has a bigger user base than other inter-book navigational features available. We believe this is because quotations themselves carry significant information and therefore are more engaging than most other link types that carry little information on the link itself.

Extracting and exposing quotations unlocks the collective effort that authors have put into identifying important and

relevant bits of information. By following the trail of quotations users are able to navigate through the collection and read perspectives from many authors on a subject they are interested in. Based on our user statistics, quotations are a promising way for people to access the links that exist in the massive collections of scanned books.

Hypertext is an enchanting device that convinces intellectually and viscerally that the act of creating and traveling connections is a part of being human. However it provides this mostly for recently authored knowledge. What new experiences might exist between the individual and the sum of human knowledge? From this research we hope to start a discussion on our relationships with old texts and the role of the old world of quotations in the new world of links.

9. ACKNOWLEDGMENTS

We'd like to thank our colleagues who provided ideas, feedback, and encouragement during this work. Adam Mathes has been a long time contributor to discussions since the very beginning, as well as UI designs. Justin Vincent carried out the user evaluations. Kathrin Paschen has been carrying out the work on the primary source identification. We appreciate the hard working students who helped us explore a number of technical issues: Andrew Gadson and Ryan Fortune.

10. REFERENCES

[1] M. Agosti and J. Allan. Special issue on methods and tools for the automatic construction of hypertext. *Information Processing and Management*, 33(2), March 1997.

[2] M. Agosti, F. Crestani, and M. Melucci. On the use of information retrieval techniques for the automatic construction of hypertext. *Information Processing and Management*, 33(2):133–144, March 1997.

[3] S. Brin, J. Davis, and H. García-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the ACM SIGMOD International Conference of Managemen of Data*, pages 398–409, San Jose, California, USA, May 1995.

[4] A. Z. Broder. On the resemblance and containment of documents. In *SEQS: Sequences '91*, 1998.

[5] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. Technical report, Digial Systems Research Center, Palo Alto, California, USA, July 1997.

[6] A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM Transcations on Information Systems*, 20(2):171–191, April 2002.

[7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[8] D. Fetterly, M. Manasse, and M. Najork. Detecting phrase-level duplication on the world wide web. In *Proceedings of the ACM SIGIR '05*, Salvador, Brazil, August 2005.

[9] R. A. Finkel, A. Zaslavsky, K. Monostori, and H. Schmidt. Signature extraction for overlap detection in documents. In *Twenty-Fifth Australasian Computer Science Conference (ACSC2002)*, pages 59–64, Melbourne, Australia, January 2002.

[10] J. Furner, D. Ellis, and P. Willett. Inter-linker consistency in the manual construction of hypertext documents. *ACM Comput. Surv.*, page 18, 1999.

[11] R. Giegerich and S. Kurtz. From Ukkonen to McCreight and Weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, 19(3):331–353, 1997.

[12] C. L. Giles, K. D. Bollacker, and S. Lawrence. Citeseer: an automatic citation indexing system. In *DL '98: Proceedings of the third ACM conference on Digital libraries*, pages 89–98, New York, New York, USA, 1998. ACM.

[13] B. Lavoie, L. S. Connaway, and L. Dempsey. Anatomy of aggregate collections. *D-Lib Magazine*, 11(9), September 2005.

[14] E. M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2):262–272, April 1976.

[15] K. Monostori, A. Zaslavsky, and H. Schmidt. Document overlap detection system for distributed digital lbraries. In *Proceedings of the ACM Digital Libraries 2000 (DL00)*, pages 226–227, San Antonio, Texas, USA, June 2000.

[16] T. H. Nelson. Xanalogical structure, needed now more than ever: Parallel documents, deep links to content, deep versioning, and deep re-use. *ACM Computing Surveys*, 31(4), December 1999.

[17] B. N. Schilit and O. Kolak. Exploring a digital library through key ideas. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '08)*, Pittsburgh, Pennsylvania, USA, June 2008.

[18] N. Shivakumar and H. García-Molina. SCAM: A copy detection mechanism for digital documents. In *Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries*, Austin, Texas, USA, June 1995.

[19] D. Sorokina, J. Gehrke, S. Warner, and P. Ginsparg. Plagiarism detection in arXiv. In *Proceedings of the Sixth International Conference on Data Mining (ICDM '06)*, Hong Kong, December 2006.

[20] A. L. Spitz. Duplicate document detection. In *Proceedings of the SPIE - International Society for Optical Engineering, Document Recognition IV*, pages 88–94, San Jose, California, USA, 1997.

[21] B. Stein and S. M. zu Eissen. Near similarity search and plagiarism analysis. In *From Data and Information Analysis to Knowledge Engineering*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 430–437. Springer, April 2006.

[22] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[23] R. Wilkinson and A. F. Smeaton. Automatic link generation. *ACM Computing Surveys*, 31(27), December 1999.