A Novel Method for Detecting Similar Documents

James W. Cooper, Anni R. Coden, Eric W. Brown
IBM T J Watson Research Center
PO Box 704, Yorktown Heights, NY 10598
{jwcnmr, anni, brown} @ watson.ibm.com

Abstract

We describe a system for rapidly determining document similarity among a set of documents obtained from an information retrieval (IR) system. We obtain a ranked list of the most important terms in each document using a rapid phrase recognizer system. We store these in a database and compute document similarity using a simple database query. If the number of terms found to not be contained in both documents is less than some predetermined threshold compared to the total number of terms in the document, these documents are determined to be very similar.

1. Introduction

One of the continuing problems in information retrieval is the fact that in the web environment, there are a large number of near-duplicate documents returned from most searches. A number of methods have been proposed for recognizing and eliminating such duplicates.

For example, Brown and Prager [1] note that documents having identical metadata such as size, date, and base filename are likely to be copies kept on different directories or on different servers and can effectively be reduced to one single occurrence.

A more sophisticated system was described by Broder [2], in which regions of each document, called "shingles" are each treated as a sequence of tokens and then reduced to a numerical representation. These are then converted to "fingerprints" using a method originally described by Rabin [3].

At a more simplistic level, Bloomfield [4] has recently described an algorithm for detecting plagiarism in which he simply searches for matches of six or more successive words between two documents.

In this paper, we explore the possibility of characterizing documents using the major terms discovered in them using phrase recognition software, and develop the hypothesis that documents which have identical lists of discovered terms are effectively identical in content.

In the first section we discuss the text mining technology we use, and how we manage the data. In the next section we discuss how we define duplicate documents and how we detect them using the results of text mining. Then in the following section, we propose a novel "document signature" for the rapid comparison of documents, and finally we discuss how this technique can be used quite successfully for finding documents that are variations on the original document.

2. The Talent Text Mining System

In approaching these document retrieval problems, we have applied a number of technologies developed in our group. In particular, we utilized the suite of text analysis tools collectively known as Talent (Text Analysis and Language Engineering Tools) for analyzing all the documents in the collection.

2.1 Textract Phrase Extraction

The primary tool we use for analyzing documents is **Textract**, itself a chain of tools for recognizing multi-word terms and proper names. We have described the features of Textract previously [5, 6]. Textract recognizes named entities [7], multi-word terms [8] and named [9] and unnamed relations between terms.

Textract reduces related forms of a term to a single *canonical form* that it can then use in computing term occurrence statistics more accurately. In addition, it recognizes abbreviations and finds the canonical forms of the words they stand for and aggregates these terms into a vocabulary for the entire collection, and for each document, keeping both document and collection-level statistics on these terms. Each term is given a collection-level importance ranking called the IQ or Information Quotient [5, 10]. IQ is effectively a measure of the document selectivity of a particular term: a term that appears in only a few documents is highly selective and has a high IQ. On the other hand, a term that appears in many documents is far less selective and has a low IQ.

IQ is measured on a scale of 0 to 100, where a value of X means that X% of the vocabulary items in the collection have a lower IQ. The IQ measure is based on the number of occurrences of a term in a collection, the number of documents the term occurs in and the number documents the term occurs in more than once. Thus, IQ is collection dependent and a term (such as "computer") that is salient in a collection of documents about entertainment might have a very low IQ in a collection of documents about computer technology.

Two of the major outputs of Textract are the IQ and collection statistics for each of these canonical terms, and tables of the terms found in each document. The terms per document are canonical forms within each document. However, additional collapsing of terms to canonical forms can also occur during a second pass at the collection level. It is the document-level terms we use in these experiments.

Textract further categorizes the entities it discovers into one of the categories shown in Table 1. The earlier categories have the least certainty and the later ones higher certainty.

Table 1 –Categories assigned to terms by Textract

UWORD	Unknown word
UTERM	Unknown term
UABBR	Unknown abbreviation
UNAME	Unknown type of name
PLACE?	Probably a place
PERSON?	Probably a person
PLACE	A place
PERSON	A person
ORG	An organization

While Textract is our tool of choice in these experiments, we note that there are a number of other systems that have been developed for phrase recognition. For example, Liddy has developed DR-LINK, which is marketed by Textwise [11], and Evans (et al) have developed LinkIT [12]. Further, somewhat similar technology is available in the ThingFinder program from InXight [13].

2.2 Characterizing Documents

We have used the Textract system in previous work to characterize documents in a number of ways. For example, we reported that Textract's term recognition combined with conventional *tf***idf* term frequency measures and some document structure information could be use to help generate fairly useful summaries of documents [14].

In a similar fashion, we discovered that we could characterize documents of little significant content using the measures generated by Textract, and eliminate them from lists of results returned from searches [15].

2.3 Data Management

Once the data from a collection of documents have been analyzed using Textract, it is useful to construct a system for managing that data in a way that makes it easy to sort the terms by document, by IQ and by term type. For this purpose, we constructed a Java class library known as KSS [16], [17] (for Knowledge System Server), which builds a database from the Textract data. You can also use these classes to create search engine indexes of the documents, and build the Context thesaurus of co-occurring terms [5].

3. What Are Duplicate Documents?

In any web search, it is fairly likely that some documents that are returned are very similar. For example, the same documents may exist on several servers or in several users' directories. One very frequent example of this are the Java API documents from Sun which are found on almost every Java developer's machine. Since these are very well known and described, they are very easy to eliminate using any of the existing techniques.

However, more difficult cases occur when

- There are several versions of the same document on various servers.
- The same document is found in several forms, such as HTML and PDF.
- A document is embedded in another. In this case the embedded document may or may not be the most significant part of the combined document.

These cases are more difficult to solve rapidly and it is these that make up the subject of this study.

For the purposes of this study, we define duplicate documents as ones that have essentially the same words in the same sentences and paragraphs. These paragraphs could be in a somewhat different order, however. The documents may be in different forms, such as HTML and PDF and they may vary slightly in header boiler-plate information.

4. Current Experiments

Given the array of sophisticated term management technologies our group has developed, we undertook to find out whether these systems can be use to detect document similarity. In particular, is it possible to use some subset of terms found by Textract as a compressed document representation, which we can use to make rapid comparisons of documents and cluster or eliminate those that are essentially identical?

4.1 Query 1: Detecting Similar Documents

In our first experiment, we went to a popular search engine site with all enhancements turned off, and issued the query "fix broken Thinkpad." This is the sort of naïve query that returns a plethora of documents the user does not want, or expect. Much as we predicted, there were *no* documents on how to repair Thinkpads. However, many of the top 50 documents contained all of these terms in some context or other. Of the top 50, we were able to download and analyze 36 of them. Ten of these documents were Adobe Acrobat PDF files. We used the Gemini plug-in [18] for Adobe Acrobat Reader to export these files into HTML.

We then created a single collection of these documents and analyzed it using Textract. Textract produces a file of

terms found in each document, and a file of all the terms in the collection along with their IQ. We used the KSS Java libraries to load these result files into DB2 and subjected the results to various SQL queries to determine the number of terms that documents had in common.

4.2 Similarity Queries

We first must find the significant terms in each document. Initially, we ranked all the terms except the unknown word types in order of decreasing IQ, and filtered to eliminate those terms which only appear once in the collection, that is those that have a frequency of 1.

The question we then want to ask, then, is which terms are not found in common between pairs of documents. You can find these in a single SQL query of the sort

```
Select count(*) as c from
  (select terms from doc1 where ..)
not in
    (select terms from doc2 where ..)
```

After some experimentation, we determined that the important selection criteria are to select terms with an IQ > 0 and which were not UWORDS. We dropped the requirement of terms having greater frequency than 1 since this made comparisons of shorter documents less accurate.

This returns the count of the number of terms that appear in document 2 that are not in document 1. While it might seem that n^2 queries are necessary, it is really only necessary to traverse the upper triangle of this matrix. We do this by sorting the documents into order of increasing size, and comparing documents with the next larger ones in the list. We further reduce the number of compares by limiting the test to documents that are no more than about 10% larger than the compared document. This parameter is, of course, adjustable.

4.3 Results

We found 6 clusters of documents in the 36 documents we analyzed in the first query. Three of these were pairs of identical documents returned from different servers, as shown in

Table 3 contains an interesting cluster of eight documents that have similar names, but different sizes. The final column of the table shows the difference in contained terms between adjacent documents in the list.

It is easy to see that these documents must be closely related versions of the same information. In fact, they are all different versions of the same IBM manual describing the Websphere server product. They differ in small details: for example one manual mentions the Linux version of Websphere and another does not. Each of these documents was returned as a PDF file and was converted to HTML using the Gemini plug-in mentioned above.

Table 2. These documents are identical by any measure and can be easily recognized and collapsed to a single entity, using the methods described by Brown and Prager [1].

Table 3 contains an interesting cluster of eight documents that have similar names, but different sizes. The final column of the table shows the difference in contained terms between adjacent documents in the list.

It is easy to see that these documents must be closely related versions of the same information. In fact, they are all different versions of the same IBM manual describing the Websphere server product. They differ in small details: for example one manual mentions the Linux version of Websphere and another does not. Each of these documents was returned as a PDF file and was converted to HTML using the Gemini plug-in mentioned above.

Table 2– Identical documents returned by Query 1

Document name	Size	# of terms
Sytpf130.html	12236	122
Sytpf1301.html	12236	122
aixy2k.html	153255	737
aixy2k1.html	153255	737
Client.html	52854	189
Conf.html	52854	189

Table 3– Very Similar Documents returned from Query 1

Number	Title	Size	Terms	Delta terms
1	Fund2	481963	2198	0
2	Fund4	481963	2207	29
3	ct7mhna1	493225	2139	0
4	Ct7mhna2	493295	2146	64
5	Fund0	503138	2235	37
6	Fund1	504737	2249	25
7	Fund3	505004	2287	11
8	Fund5	505471	2271	--

In Table 3, documents 1 and 2 and documents 3 and 4 are almost certainly absolutely identical. However, the remaining four documents are clearly all closely similar as well. This algorithm finds such documents even when simpler methods would not.

We note that one could remove some document from the cluster of very similar documents if the terms that are

different between them include a term that is also contained in the original query.

This search also returned two other closely related document pairs as shown in Table 4. Documents 9 and 10 are in fact a draft and a final PDF document of a paper by Selker and Burleson. [19] Since these papers are quite different in size and format, they would probably not have been found as similar by other previously described methods. The term differences between the two are partly because of some additional abstract and polishing, and partly because of the included boilerplate from the magazine format.

Documents 11 and 12 in Table 4 are much less closely related by inspection, although the number of terms they have in common is quite high. These are in fact a false match that we generated by eliminating both unknown words and unknown names from the comparison. Both documents are, in fact, articles about software updates for Thinkpads, but one is about video features for Windows 3.1 on a Thinkpad 380 and 600, and the other about the latest video driver for OS/2 for the same two machines. If we do not exclude unknown names (which in this case are part and model numbers) the documents are not suggested as similar.

Table 4 – Pairs of Similar Documents returned by Query 1

#	Title	Size	Terms	Delta
9	Selker3.htm	50169	257	23
10	Selker.htm	54274	218	--
11	Manager.htm	15903	91	8
12	Manager1.htm	16000	80	--

5. Detecting Identical Documents

When documents are very large, it is not usually convenient to run phrase recognition software on the entire set of documents in real time when they are returned from a query, because the elapsed time is too great. However, as part of the indexing process, it is not unreasonable to catalog the major terms in each document. However, even making comparisons among large numbers of terms in multiple documents can take many seconds and can lead to unacceptable delays in returning answers.

We suggest that it is possible to compute a digital signature of each document, based on the terms we find in it. Such a signature can be as simple as a sum of the hash codes of the term strings that make up these major terms. In this experiment, we used the Java String class's hashCode method to compute a code for each term found in a document, and then added these codes up to form the signature. The results are shown in Table 5. The number suffixes are used to indicate identical url names from different servers.

Consulting Table 5, it is clear that even though documents 1 and 2 and documents 3 and 4 have similar names and identical sizes, they are not exactly the same, since the signatures differ. On the other hand, documents 13 and 14 are identical, as are documents 15 and 16 and documents 17 and 18. To validate this computation, we ran a query to find which terms actually appear in document 2 that do not appear in document 1. These are shown in Table 6.

We note that two documents could be considered identical by this procedure if they contained the same paragraphs in a different order, or even the same sentences in a different order

Table 5 – Computed document signatures for similar documents from Query 1.

Number	Url	Size	Signature
1	Fund2	481963	24681058826
2	Fund4	481963	26371644438
3	Ct7mhna1	493225	33130506660
4	Ct7mhna2	493295	32920545738
5	Fund0	503138	10451017932
6	Fund1	504737	8933271588
7	Fund3	505004	7211814280
8	Fund5	505471	12114748848
13	Sytpf130	12236	13802917585
14	Sytpf1301	12236	13802917585
15	aixy2k	153255	-28232730155
16	aixy2k1	153255	-28232730155
17	Client	52854	6580188642
18	Conf	52854	6580188642

Table 6 –Terms found in Document 2 but not in Document 1

Current information
Database cleanup utility
Marketing campaign
PDF document
Product attribute
Shopper request

.

We further note that while individual strings will usually have unique hash codes, there is a somewhat larger probability that the sum of a series of hash codes will be less unique. However, the probability of these collisions is small enough that these document signatures remain quite useful. Further, it is even less likely that documents with accidentally identical signatures would be returned from a query if they were not the same document.

To summarize, these document signatures simply provide a shorthand method of representing the top terms in documents, so that they can be compare very rapidly. The actual technique for comparison is essentially the same as in section 4.1.

6. Query 2 – Smaller Documents

In a second series of experiments, we issued a more focused query "Program ViaVoice in Java," and were able to retrieve 47 of the top 50 returned documents. Since many of these had the same filename, we carefully renamed them when we save the local copies for analysis.

Since all of these documents were of modest size (the top one was 75 K) we found that we could perform the entire analysis on the documents fast enough that it could be carried out more or less in real time in response to a query.

The results included 8 pairs of identical documents as measured by size and the signature we described above. In addition, the results contained the 13 very similar documents shown in Table 7.

Table 7 –Very Similar Documents Returned by Query 2

#	# Terms	delta	Url	Size	Signature
23	47	1	News11	37788	-9902679640
32	47	1	News9	38100	**-9692972733**
28	48	0	News5	38383	-11688727862
24	47	1	News12	38604	**-9692972733**
31	48	0	News8	38727	-9921821918
25	47	0	News2	39389	**-9692972733**
29	47	0	News6	39389	**-9692972733**
26	47	0	News3	39465	**-9692972733**
27	47	1	News4	39465	**-9692972733**
19	46	1	News0	39580	-5116033555
21	46	3	News1	39580	-8166342237
22	47	1	News10	40537	-11188846377
30	48	--	News7	40537	-12715476873

The documents in **Table 7** are all very similar, since they differ in only 1 or 2 terms out of 47, and all have similar sizes. Based on size alone, you would identify only 4 pairs of identical documents. However, all of these are detected as similar based on the fact that they contain the same terms. In addition, it is significant that six of these documents have identical signatures (shown in boldface) even though they are of four difference sizes. This shows the power of the signature method for rapid identification of documents. For any new document, you can compute its signature and quickly compare it with other documents signatures. If it is identical, you can also compute whether these documents contain similar terms.

7. Finding Similar Documents

In the foregoing, we have discussed the problem of finding very similar documents, in most cases so similar that only one of them need be returned from a search at all.

There is another set of problems to solve, however, relating to documents that are similar because they exist in a number of revisions. In order to test this algorithm for this new problem, we determined that we should relax the restrictions regarding the percentage of terms that could be different, and the size differences we would allow between documents we compare.

In the first experiment, we collected 13 documents about IBM financial and banking products and services from the ibm.com web site, so that they would all have a relatively similar vocabulary, and one document which was a press release about IBM's performance in a supercomputer comparison. We expected that this last document would have a markedly different vocabulary from the others.

Then we took this last supercomputer document and cut out a 2533 byte segment comprising the main news story without any of the surrounding HTML formatting, and pasted it into each of the other financial documents. We then ran Textract and indexed the terms per document as described above and ran the same experiment on document similarity, where we changed the SQL query to allow the fraction of different terms to be as high as 0.5. This query identified every document pair correctly and did not find any pairs of documents similar except those consisting of an original and the same document with inserted text.

Table 8 shows the fraction of terms that differ between the original document and the same document when the news release text is inserted. The fractional differences vary between 0.01 and 0.157.

Table 8 – Fractional differences in terms in financial documents when an news release on supercomputers was added to each of them

#	Original url	Fraction of different terms with inserted news release
21	Folder1	0.100
38	Reuters	0.157
30	Nletter	0.06
20	Folder	0.117
17	Ebus3	0.076
16	Ebus1	0.055
35	Retaildel	0.096
27	Kookmin	0.054
4	CRM	0.100
1	24552	0.040

8	Building	0.040
14	Ebusmark	0.010
33	RetailB	0.015

We concluded that documents that had less than 20% of the terms different were likely to represent documents that were related and contained much the same text. In fact, we found it quite encouraging that this method identified every such document correctly and returned no false positives. (A false positive would be a difference in terms of less than 20% in documents that were in fact different.) In other words the precision and recall were 100%.

On the other hand, the algorithm did not identify the short IBM press release document as being related to any of the others by containment, since it was relatively short, and contained fewer salient terms.

7.1 Inserting Similar Document Text

In this experiment, we took one of the financial documents, called CRM in

Table 8, and inserted 3276 bytes of it, comprising nearly all of the non-markup text, into all of the other documents in the set. We found much the same results as above: 100% precision and recall as shown in Table 9.

Again, all of the documents with the inserted text were detected as similar to the originals and no false positives were detected. The fraction of terms that were different was 0.125 or under, except for the case where the larger CRM document was added to the smaller Ibmtop document.

Table 9 - Fractional differences in terms in financial documents when a similar article was added to each of them.

Doc number	Original url	Fraction different with CRM inserted in it	Fraction different between CRM added and ibmtop added
21	Folder1	0.100	0.050
38	Reuters	0.111	0.111
30	Nletter	0.064	0.064
20	Folder	0.125	0.125
17	Ebus3	0.040	0.040
16	Ebus1	0.055	0.055
35	Retaildel	0.066	0.066
27	Kookmin	0.027	0.027
1	24552	0.080	0.020
8	Building	0.125	0.104
14	Ebusmark	0.025	0.020
33	RetailB	0.020	0.020
24	Ibmtop	0.500	---

In the final test, we ran both experiments simultaneously, and found that all the similar documents were detected as before. In addition, in an unexpected result, all of the documents with the Ibmtop text were found to be similar to the corresponding document with the inserted CRM text as well. This is shown in the fourth column of **Table 9**.

8. Implementation Details

In these experiments, we ran the Textract text mining program on the collection of documents (around 50) returned from the query. Then we generated low-level DB2 table load files [20] from the Textract output and loaded the terms/document data into DB2. The IQ and frequency of the terms was determined from this collection. Thus, IQ would change somewhat based on the contents of the documents returned. A term that was highly salient in one document set might appear too frequently to be very selective in another set. However, we have eliminated much of this dependence by simply requiring that the IQ value be non-zero. It would in general be possible to maintain a vocabulary for a search system with IQs predetermined.

When all of the documents are relatively short, it is quite possible to do this more or less in real time. However, when longer documents make the mining processes too slow, it is necessary to index and mine the documents in advance and cache the results, just as you do with the document search indexes. When database comparisons of very long documents are too slow, it is possible to just compare the top terms, for example the top 200 terms in each document. Finally, it is quite reasonable to store the document signature we describe as part of the database document table, so you can compare documents quickly.

In the course of these experiments, we varied the IQ threshold and the term frequency threshold. For various types of applications, these values may well need to be adjusted. However, it is important to note that the document signature is dependent on the number of terms you retrieve, and if you change your criteria, you will need to recompute these signatures.

In comparing documents for close similarity as we did in Query 1, we only considered documents that were within 10% of the size of the one we were comparing to, and only considered documents to be similar when the number of terms that were different was less than 10% of the total number of terms in the smaller document. In comparing documents that contained embedded additional material, we relaxed both of these criteria to 50%, with little performance penalty.

In doing the database comparisons we discuss in 4.2, the SQL query could undoubtedly be speeded up if the terms were compared by integer key instead of using a string compare. This and similar enhancements are the subject of further work. In addition, we will be experimenting with different percentages of changes in

documents to see how different they can be and still be recognized as similar.

9. Summary

We define similar documents as ones that have essentially the same sentences and paragraphs, but not necessarily in exactly the same order. We have found that we can accurately compute whether documents are similar by comparing the number of terms found using a phrase recognition program such as Textract.

We further found that you can accurately recognize documents that have been revised to contain parts of other documents as still being closely related to the parent document. Finally, we described a novel document signature that you can use to make a rapid comparison between documents that are likely to be identical.

This method has some superficial similarity to the "shingles" approach [7], but, pending testing with shingles, it is presumably an improvement because shingles are typically larger groups of terms. Our method uses only the salient terms to characterize documents, and these terms can appear in a different order and still provide the same characterization of the document. Further, phrase recognition programs such as Textract generally reduce the found terms to a root or "canonical" form, so that even if the terms appear in different variant forms in slightly edited versions of a document, they will be recognized as being the same root term and found to be identical. Finally, this method is insensitive to the addition of additional polishing sentences or the rearrangement of whole paragraphs in edited versions of a document.

This system has broad applicability in improving the results of searches of large document collections, whether the returned documents have been indexes for their term content in advance or not. It can also be used for rather sophisticated plagiarism detection, or as an adjunct in finding further documents of interest and grouping these documents for the user's convenience.

10. References

[1] Brown, Eric W. and Prager, John M., US Patent 05913208.

[2] Broder, Andrei Z, "Identifying and Filtering Near-duplicate Documents," *Combinatorial Pattern Matching, 11th Annual Symposium, Montreal, Canada, June, 2000.*

[3] Rabin, M. O., "Fingerprinting by random polynomials, " Center for Research in Computing Technology, Harvard University, Report TR-15-81, 1981.

[4] Bloomfield, Louis, University of Virginia, interviewed on NPR's *All Things Considered*, May 9, 2001. See www.plagiarism.phys.virginia.edu.

[5] Cooper, J. W. and Byrd, R J, "Lexical Navigation: Visually Prompted Query Refinement," ACM Digital Libraries Conference, Philadelphia, 1997.

[6] Cooper, James W. and Byrd, Roy J., OBIWAN – "A Visual Interface for Prompted Query Refinement," Proceedings of HICSS-31, Kona, Hawaii, 1998.

[7] Ravin, Y. and Wacholder, N. 1996, "Extracting Names from Natural-Language Text," IBM Research Report 20338.

[8] Justeson, J. S. and S. Katz "Technical terminology: some linguistic properties and an algorithm for identification in text." *Natural Language Engineering*, 1, 9-27, 1995.

[9] Byrd, R.J. and Ravin, Y. Identifying and Extracting Relations in Text. *Proceedings of NLDB 99*, Klagenfurt, Austria.

[10] Prager, John M., Linguini: Recognition of Language in Digital Documents, *in Proceedings of the 32nd Hawaii International Conference on System Sciences*, Wailea, HI, January, 1999.

[11] Mnis-Textwise Labs, www.textwise.com. DR-LINK was developed at Syracuse University and is marketed by Textwise.

[12] Evans, D. K., Klavans, J. and Wacholder, N., "Document Processing with LinkIT," *Proc. Of the RIAO Conference*, Paris, France, 2000.

[13] InXight, Inc. www.inxight.com

[14] Neff, Mary S. and Cooper, James W. "Document Summarization for Active Markup," *in Proceedings of the 32nd Hawaii International Conference on System Sciences*, Wailea, HI, January, 1999.

[15] Cooper J.W. and Prager, John M. "Anti-Serendipity – Finding Useless Documents and Similar Documents," *Proceedings of the 33rd Hawaii International Conference on System Sciences, Maui, HI*, January, 2000.

[16] Cooper, J. W. "The Technology of Lexical Navigation," Workshop on Browsing Technology, *First Joint Conference on Digital Libraries*, Roanoke, VA, 2001.

[17] Cooper, J.W., Cesar, C., So, Edward, and Mack R. L., "Construction of an OO Framework for Text Mining," *OOPSLA*, Tampa Bay, 2001.

[18] Gemini plug-in for Adobe Acrobat Reader, Iceni Technology, Ltd, Norwich, England, www.iceni.com.

[19] Selker, T. and Burleson, W. "Context-aware Design and Interaction in Computer Systems," *IBM Systems Journal*, **39,** 891 (2000).

[20] Cooper, J W, "Loading Your Databases," *JavaPro*, May, 2000.