Organizing News Archives by Near-Duplicate Copy Detection in Digital Libraries

Hung-Chi Chang[1] and Jenq-Haur Wang[2]

[1] Institute of Information Science
Academia Sinica, Taiwan
[2] Department of Computer Science and Information Engineering
National Taipei University of Technology, Taiwan
hungchi@iis.sinica.edu.tw, jhwang@csie.ntut.edu.tw

Abstract. There are huge numbers of documents in digital libraries. How to effectively organize these documents so that humans can easily browse or reference is a challenging task. Existing classification methods and chronological or geographical ordering only provide partial views of the news articles. The relationships among news articles might not be easily grasped. In this paper, we propose a near-duplicate copy detection approach to organizing news archives in digital libraries. Conventional copy detection methods use word-level features which could be time-consuming and not robust to term substitutions. In this paper, we propose a sentence-level statistics-based approach to detect near-duplicate documents, which is language independent, simple but effective. It's orthogonal to and can be used to complement word-based approaches. Also it's insensitive to actual page layout of articles. The experimental results showed the high efficiency and good accuracy of the proposed approach in detecting near-duplicates in news archives.

Keywords: Near-duplicate document copy detection, sentence-level features, news archive organization.

1 Introduction

There are huge numbers of documents in digital libraries, for example, academic papers, news archives, and historic documents, to name a few. How to effectively organize these documents so that humans can easily browse or reference is a challenging task. Several existing methods might help better organizing them. For articles in news archives, they might be related to specific named entities such as people, event, time, location, and objects. Named entity recognition or extraction methods can extract these named entities with good accuracy. Text classification or clustering approaches could be used to effectively group news articles with similar semantic content into thematic topics. However, these classification methods in different dimensions only provide partial views of the news articles. For example, the relationships among news articles might not be easily grasped using these methods. Chronological ordering of news articles might organize according to their time of writing or archiving. Geographical information such as the place of writing or archiving provides another good

D.H.-L. Goh et al. (Eds.): ICADL 2007, LNCS 4822, pp. 410–419, 2007.

source of clues. However, time and location information may not be readily available if not explicitly specified in the news articles or their metadata. They are hard to pinpoint without further analysis of their relationships with other articles that have explicit time/location information.

In this paper, we propose a different approach to organizing news archives in digital libraries. First, near-duplicate copy detection techniques are applied to articles in news archives. Features are extracted and similarity among documents is estimated. Then, the candidate lists of near-duplicates in the archive form several clusters of documents which might be combined with above-mentioned classification methods as an alternative way to organize them.

Conventional document copy detection methods deal with the problem in two general word-based approaches. First, exact or "almost identical" copies can be detected with *document fingerprinting* approaches. Second, similar or "relevant" documents can be identified using *information retrieval* approaches based on bag-of-word frequencies. In the growing applications in Web community such as blogs, partial or "subset" copies are more common than exact copies of whole documents. Conventional near-duplicate document copy detection methods are usually based on different types of content editing behaviors. For example, several subcategories of near-duplicate copy detection are defined [12] based on the editing styles such as block adding/deletion, minor change, minor change and block edit, block reordering, bag-of-word similarity, and exact copy. In this paper, we intend to focus on efficiently detecting near-duplicates based on the editing results of documents.

Several issues have to be addressed. First, conventional word-based features cannot be directly applied since the word features in an extracted text segment might be quite different from those in a target document. Second, efficiency is critical in real applications. Word-based approach usually requires comparison of word *n*-grams in source document with all candidates in the target documents to find all possible matches, which is time-consuming. Third, most word-based approaches try to speed up the matching process by using "hashed *n*-grams" or shingles [2]. However, the accuracy of detection could be significantly influenced if some terms were substituted in the copied segments.

In this paper, we propose an effective sentence-level approach that exploits features beyond word-level features. We try to somehow capture part of the writing style above the word semantics. The most representative sentence-level statistics are extracted from the documents, the *sentence length*. It's the most primitive syntactic information about a document which could vary in different people's writing styles. One important advantage of the approach is its independence of the specific language used in writing. Since only the sentence delimiter will be used in determining the boundaries, it's insensitive to actual article layout. Also, it's both efficient and effective in detecting partial copies of text segments. The promising experimental results showed high accuracy of detection and good efficiency of our proposed approach.

The rest of the paper is organized as follows. Section 2 lists some related work. In Section 3, the details of the proposed approach are illustrated. Section 4 presents the experiments and potential applications of the approach. In Section 5, we list the conclusions.

2 Related Work

The research field of document copy detection has received much attention thanks to many popular applications on the Web, for example, duplicate Web page detection [5] and removal in Web search engines, document versioning and plagiarism detection [6] in digital libraries, and duplicate document removal in databases [1][9]. Conventionally, there are two general approaches to near-duplicate document copy detection: document fingerprinting and bag-of-word similarity. Document fingerprinting approaches [4] can detect almost-identical documents with hash-value based checksum or "fingerprint". Earlier copy detection systems such as SCAM [9] and COPS [1] are some examples. Bag-of-word similarity approaches try to identify relevant documents with similar distribution of word frequencies.

New applications in Web community such as weblogs and wikis raise new challenges to the existing approaches. First, there are more partial or "subset" copies than exact copies of the whole document. Copy detection techniques dealing with whole document matching might not be directly applicable to the partial copy cases. There are similar efforts in detecting documents with "intermediate-level similarity" between the semantic relevance and syntactically identical. For example, Shulman [10] targeted at the task of near-duplicate detection in notice and public comment rulemaking. Yang and Callan [12] proposed an instance-level constrained clustering approach that incorporates additional information such as document attributes and content structure into the clustering process to form near-duplicate clusters.

3 The Proposed Approach

The basic idea of the proposed approach is to use sentence-level features, i.e. *sentence length*, as an alternative to word-based features. The system architecture of the main functional blocks is illustrated in Fig. 1.

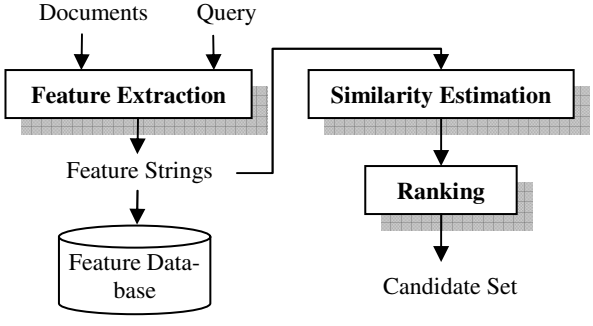

Fig. 1. Architecture of the main functional blocks

The source data is first converted into a *feature string*, a numeric sequence representing the number of words in sentences, by the *feature extraction* module. Some pre-processing steps such as HTML markup removal and stopword removal might be

applied first. The *similarity estimation* module then extracts *feature vectors* from the feature string and indexes them into the feature database which records the (Feature vector, Document ID) pairs.

For a given query, the same feature extraction procedure is used to generate a query feature string. The feature vectors extracted from the querying feature string are then used to search the feature database. A *ranking* scheme would then be applied to order the candidate set of query results by their similarity scores.

3.1 Feature Extraction

In feature extraction module, text-based sentences are determined by the pre-defined delimiter set. A string of text would be considered a single sentence if and only if it satisfies both conditions: (1) the string is between two arbitrary symbols defined in the delimiter set; and (2) it does not contain any symbol defined in the delimiter set. For example, punctuation marks such as "." and "!" are included in the delimiter set. Delimiters should be chosen carefully, because it would influence the result of conversion and also the quality of subsequent functions. The output of feature extraction would be a numeric sequence, the *feature string* for the document. Further processing would be done on the feature string instead of the original text content.

3.2 Similarity Estimation

In order to detect partial document copies, a sliding window is used to extract feature vectors, the sub-sequences from the original feature strings, for similarity estimation. There are two major parameters: the window size WS and the sliding step width SW. Window size defines the constant length of the current feature vectors that we're dealing with, and sliding step width defines the number of elements to slide through at each step. It is referred as jumping window if SW > 1 [13]. For static jumping, SW is constant during indexing. For example, for feature string [0123456789…] (with WS=4 and SW=2), the first two extracted feature vectors would be [0123] and [2345] respectively. The effect of varying window size and sliding step width is listed in Fig. 2. We would verify these effects in later section.

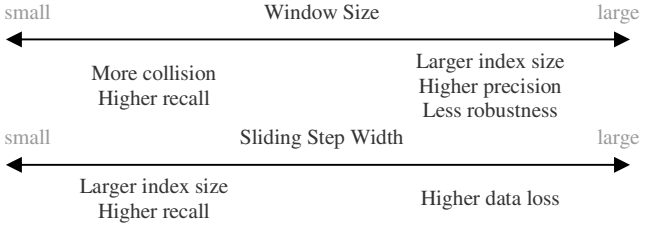

Fig. 2. The effect of window size and sliding step width

Considering the case when the feature strings are `[0123456789…]` and `[1234567890…]` respectively. It should be identified as near-duplicate for the matched segment `[123456789]`. However, in previous example of static jumping (where WS=4 and SW=2), we will not be able to find any match between the feature

vectors extracted from the two feature strings, since none of the respective features vectors `[0123]`, `[2345]`, `[4567]`, ..., and `[1234]`, `[3456]`, `[5678]`, …, will match. To avoid such situation due to the jumping in feature string, we only applied static jumping window in indexing stage to reduce the database size in our implementation.

Instead of indexing the extracted feature vector in each sliding step, another scheme called *dynamic jumping* might be applied to decrease the number of indexed records and to reduce the storage space usage. The idea of dynamic jumping is described as follows: If the leading SW elements (E_n, E_{n+1}, …, E_{n+SW-1}) at the current window position are equal to the SW elements at the previous window (E_{n-SW}, E_{n-SW+1}, …, E_{n-1}), that is, the SW elements in front of the current window, and also equal to the SW elements at the next window (E_{n+SW}, E_{n+SW+1}, …, $E_{n+2SW-1}$), then *discard* the current window since the repeated pattern has been extracted in the adjacent vectors.

When determining if two numeric sequences are near-duplicates, we set a *minimum matching threshold* t, which controls the minimum number of contiguously matched sub-sequences before being considered as near-duplicates. The *minimum matching length* could be expressed as follows:

$$Length_{MIN\_Match} = WS + SW * (t - 1). \tag{1}$$

When t=1, we're only considering the effect of window size for determining if two sequences are near-duplicates. Assume that there are two configurations with different windows sizes $WS_L > WS_S$, step widths SW_L and SW_S, thresholds t_L and t_S, respectively. If we fix the minimum matching length, then we have:

$$Length_{MIN\_Match} = WS_S + SW_S * (t_S - 1) = WS_L + SW_L * (t_L - 1). \tag{2}$$

With the constraint of fixed index size, we would get that $SW_L > SW_S$. The relationship among these arguments is shown as Fig. 3.

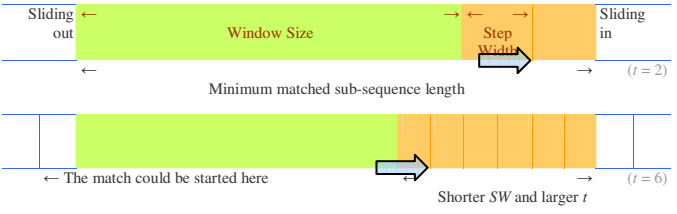

Fig. 3. The relationship among parameters. The top half shows the case of larger window size WS_L while the bottom half shows the smaller windows size WS_S.

From equation (2), we could see that a configuration with smaller window size is more flexible and might achieve higher recall than the one with larger window size because shorter SW implies that the matched sub-sequence or feature vector could start at more points. The case of smaller windows size also preserves more details about original feature string because of shorter SW. However it requires more computation because $t_S > t_L$.

3.3 Ranking Scheme

After similarity estimation, there could be more than one document matching feature vectors extracted from a given query Q. We assume that the more matches of a document

by a feature vector in Q, the more relevant the query is. Then, the similarity score of a document to a query Q would be calculated based on its weighted sum of the number of matched feature vectors. That is, a document is more likely to contain duplicated content with the query if it has a higher score. Finally, we could rank the documents in the candidate set according to their scores.

4 Experiments

The experiments are divided into two parts. First, the effects and relationship among various parameter configurations were evaluated. Then, the performance of different matching and ranking schemes were compared. All experiments were carried out on the same desktop PC with Intel Pentium 4 3.0 GHz CPU and 2 GB DDR2 RAM.

4.1 Parameter Fine-Tuning

The first experiment was conducted to evaluate the effects and relationship among parameter configurations. We first collected the proceedings of WWW and SIGIR conferences from 2004 to 2006, a total of 1,090 papers in PDF format. These papers were converted to text files to form the *source data set*. Then, we selected the 27 nominated best paper award candidates and extracted their introduction and conclusion sections into a single query file as the *test query*. For each selected paper, this step could be regarded as the copying and editing behaviors; while the query file could be considered as a near-duplicate of each selected paper. We tested our proposed approach using 15 different parameter configurations as listed in Table 1.

Table 1. Various parameter configurations and their index statistics

Notation	*WS*	*SW*	*DJ*	Index number	Index time (sec.)
WS08SW1	8	1	No	545,254	2.233
WS08SW1wDJ	8	1	Yes	508,468	2.096
WS08SW2	8	2	No	273,442	0.984
WS16SW1	16	1	No	536,642	2.475
WS16SW1wDJ	16	1	Yes	501,422	2.310
WS16SW2	16	2	No	269,135	0.951
WS16SW4	16	4	No	135,386	0.404
WS32SW1	32	1	No	519,426	3.132
WS32SW1wDJ	32	1	Yes	486,835	2.716
WS32SW4	32	4	No	131,082	0.479
WS32SW8	32	8	No	66,338	0.209
WS64SW1	64	1	No	485,134	3.457
WS64SW1wDJ	64	1	Yes	456,995	3.121
WS64SW8	64	8	No	62,039	0.310
WS64SW16	64	16	No	31,827	0.139

Note that, for each *WS* setting, we only applied the *dynamic jumping* (denoted as *DJ* in the table) scheme to the case *SW*=1 for comparison. Since the purpose of these experiments was to evaluate the effect of various parameter configurations, no ranking were done here. The matched documents by any feature vector extracted from the query file would be added to query result. As shown in Table 1, we could see that the

dynamic jumping scheme reduced the number of indexed feature vectors by around 7% and index time by around 10% of the original in indexing stage. Under these configurations, we tested the query performance in terms of the efficiency and effectiveness as shown in Tables 2 and 3.

Table 2. Search time in querying stage

Configuration	Search time (sec.)	Configuration	Search time (sec.)
WS8SW1	6.469	WS32SW1	0.344
WS8SW1wDJ	1.766	WS32SW1wDJ	0.297
WS8SW2	3.204	WS32SW4	0.141
		WS32SW8	0.078
WS16SW1	0.219	WS64SW1	0.125
WS16SW1wDJ	0.172	WS64SW1wDJ	0.125
WS16SW2	0.156	WS64SW8	0.078
WS16SW4	0.109	WS64SW16	0.062

Table 3. Query result of various configurations

Configuration	Recog.	Total	Recall	Precision	F1
WS8SW1	26	570	0.963	0.046	0.087
WS8SW1wDJ	26	353	0.963	0.074	0.137
WS8SW2	26	459	0.963	0.057	0.107
WS16SW1	24	24	0.889	1.0	**0.941**
WS16SW1wDJ	24	24	0.889	1.0	**0.941**
WS16SW2	24	24	0.889	1.0	**0.941**
WS16SW4	23	23	0.852	1.0	0.920
WS32SW1	12	12	0.444	1.0	0.615
WS32SW1wDJ	12	12	0.444	1.0	0.615
WS32SW4	12	12	0.444	1.0	0.615
WS32SW8	12	12	0.444	1.0	0.615
WS64SW1	1	1	0.037	1.0	0.071
WS64SW1wDJ	1	1	0.037	1.0	0.071
WS64SW8	1	1	0.037	1.0	0.071
WS64SW16	1	1	0.037	1.0	0.071

We use the term *recognition* to denote the number of selected papers found in query result. We could see that the dynamic jumping scheme improved the efficiency without losing the effectiveness of the query result. We could also find that recall dropped dramatically as the window size increased. Nevertheless, precision reached 1.0 when window size is larger than or equal to 16. This observation is encouraging that the proposed approach is discriminative enough with low cost.

Note that one of the selected papers could not be matched by all queries. We checked this paper and found that the introduction and conclusion of the poster paper are too short. There are also problems for converting PDF to text file, such as unexpected hyphenation to break single words into two. This requires manual processing before further experiments can be done.

4.2 Evaluation on Similarity Estimation and Ranking

When we adopted a smaller window size (WS=8), the precision was quite low as in Table 3. The reason could be lots of false positives incurred by matching smaller

fragment of numeric string. Thus, we further applied our ranking scheme to improve the precision. As shown in Table 4, we simply compared the top 27 of the ranked list with the original result without ranking. It could be found that our ranking scheme helped to achieve high precision without sacrificing much recall.

Table 4. Query performance with and without ranking

Configuration	Recog.	Total	Recall	Precision	F1
WS8SW1wDJ	26	353	0.963	0.074	0.137
WS8SW1wDJ, w/ ranking, top 27	25	27	0.926	0.926	0.926

The second experiment was conducted on a larger data set to evaluate our similarity estimation and ranking schemes. The source data set is shown in Table 5. We selected six English document collections from NTCIR-4 Test Collections for CLIR task [7]. 30 documents from each collection, that is, a total of 180 documents, were randomly selected as the test queries. Then we manually composed the query file in the same manner as the previous experiments. At most 10 paragraphs per document (about 6 paragraphs per document on the average) were copied. Note that documents in this dataset (news articles), were much shorter in general and more varied in length than the previous one (academic papers).

Table 5. Source data set

Document collection	# of doc	Size (MB)
EIRB010 (Taiwan News and Chinatimes English News, 1998 - 1999)	10,204	24.6
Hong Kong Standard (1998 – 1999)	96,856	253.2
Korea Times (1998 – 1999)	19,599	55.9
Korea Times (2000 – 2001)	30,530	81.1
Mainichi Daily News (1998 – 1999)	12,723	33.3
Mainichi Daily News (2000 – 2001)	12,155	26.3
Total	182,067	474.4

The feature database in this experiment was stored on disk instead of in memory because it's much larger in size. The query data set was indexed with two configurations: WS8SW1 (*WS*=8 and *SW*=1) and WS16SW2 (*WS*=16 and *SW*=2). To concentrate our attention on the performance of ranking scheme, dynamic jumping was disabled in both configurations. Summary of index statistics for these two configurations is listed in Table 6. It took much longer to index because of the extra time requirement introduced by disk access. The performance comparison among configurations with various *minimum matching threshold t* is shown in Table 7.

Table 6. Summary of index statistics

Metrics	WS8SW1	WS16SW2
Index number	5,356,152	2,195,642
Index time (second)	2,100.49	274.58

Table 7. Performance comparison among configurations with various thresholds *t*

Configuration	Time	Recog.	Total	R	P	F1
WS8SW1, *t*=1	0.156	178	230	**0.989**	0.774	0.868
WS8SW1, *t*=3	0.172	175	198	0.972	0.884	0.926
WS8SW1, *t*=5	0.188	163	168	0.906	0.970	**0.937**
WS8SW1, *t*=9	0.188	143	146	0.794	0.979	0.877
WS16SW2, *t*=1	0.094	139	141	0.772	**0.986**	0.866

R: Recall, **P**: Precision

For different thresholds *t*, recall decreased and precision increased as *t* increased. Comparing the case *t*=1 of WS16SW2 with the case *t*=9 of WS8SW1 in Table 7, which is an example of Fig. 3, we verified that smaller windows size was more flexible and achieved higher recall than larger ones under the assumptions of invariant minimum matching length and constant index size. However, it would take longer time for query. There is a tradeoff between efficiency and efficacy.

4.3 Possible Applications

The motivation of this work is to detect quotation without explicit reference to the original news articles. It could be applied in other situations. For example, when a blogger is posting a new article on the blog site, the system checks the submitted content against the protected archive. If any matched article is found, notification would be sent to the posting user and the authors of matched candidates.

In addition to triggering the detection by a query document, for some applications, we have only one data set, for example, the collection of Web pages. The proposed approach could be easily modified to handle such case. Instead of extracting and indexing feature vectors from feature string of each document, similarity scores among documents would be calculated. Then, documents could be clustered into groups based on the similarity scores and clustering policy.

Link analysis has been proven effective for many Web applications, such as Web page ranking and classification. However, explicit hyperlinks do not exist among news articles. One possible solution is to introduce an alternative resource of links among articles, that is, implicit links [8][11]. Under the assumption that documents with the same duplicated content could be regarded relevant to each other, our proposed approach could be modified to identify this kind of implicit links.

5 Conclusions

In this paper, we propose a sentence-level statistics-based near-duplicate copy detection approach to organizing news archives in digital libraries. The proposed approach is language-independent, simple but effective. It is orthogonal to and can be used to complement conventional word-based approaches. The experimental results show the potential of its effectiveness and efficiency in near-duplicate document copy detection for news archive. One possible limitation of the approach is that it could not effectively handle the well-structured syntactic paragraphs with repetitive or regular sentence length pattern, for example, Tang poetry or lyrics, since their sentence lengths carry not much information about the document content. Further investigation is needed to exploit such special application in copy detection.

References

1. Brin, S., Davis, J., Garcia-Molina, H.: Copy Detection Mechanisms for Digital Documents. In: ACM SIGMOD International Conference on Management of Data, pp. 398–409 (1995)
2. Broder, A., Glassman, S., Manasse, M., Zweig, G.: Syntactic Clustering of the Web. In: 6th International World Wide Web Conference, pp. 393–404 (1997)
3. Charikar, M.S.: Similarity Estimation Techniques from Rounding Algorithms. In: 34th Annual ACM Symposium on Theory of Computing, pp. 380–388 (2002)
4. Heintze, N.: Scalable Document Fingerprinting. In: Proceedings of the 2nd USENIX workshop on Electronic Commerce, pp. 191–200 (1996)
5. Henzinger, M.: Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms. In: Proceedings of SIGIR 2006, pp. 284–291 (2006)
6. Hoad, T.C., Zobel, J.: Methods for identifying versioned and plagiarized documents. Journal of the American Society for Information Science and Technology 54(3), 203–215 (2003)
7. NTCIR (NII Test Collection for IR Systems) Project, http://research.nii.ac.jp/ntcir/
8. Shen, D., Sun, J.T., Tang, Q., Chen, Z.: A Comparison of Implicit and Explicit Links for Web Page Classification. In Proceedings of WWW 2006, pp. 643–650 (2006)
9. Shivakumar, N., Garcia-Molina, H.: SCAM: a copy detection mechanism for digital documents. In: Proceedings of International Conference on Theory and Practice of Digital Libraries (1995)
10. Shulman, S.: E-Rulemaking: Issues in Current Research and Practice. International Journal of Public Administration 28, 621–641 (2005)
11. Xu, G., Ma, W.Y.: Building Implicit Links from Content for Forum Search. In: Proceedings of SIGIR 2006, pp. 300–307 (2006)
12. Yang, H., Callan, J.: Near-Duplicate Detection by Instance-level Constrained Clustering. In: Proceedings of SIGIR 2006, pp. 421–428 (2006)
13. Zhu, Y., Shasha, D.: StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In: Proceedings of the 28th ACM VLDB International Conference on Very Large Data Base, pp. 358–369 (2002)