Efficient Approach for near Duplicate Document Detection using Textual and Conceptual based Techniques

Rajendra Kumar Roul[1], Sahil Mittal[2], Pravin Joshi[2]

[1,2] BITS Pilani K. K. Birla Goa Campus, Zuarinagar, Goa-403726, India
[1]rkroul@goa.bits-pilani.ac.in
[2]{sahilindia33, pravinjoshi95}@gmail.com

Abstract. With the rapid development and usage of World Wide Web, there are a huge number of duplicate web pages. To help the search engine for providing results free from duplicates, detection and elimination of duplicates is required. The proposed approach combines the strength of some "state of the art" duplicate detection algorithms like Shingling and Simhash to efficiently detect and eliminate near duplicate web pages while considering some important factors like word order. In addition, it employs Latent Semantic Indexing (LSI) to detect conceptually similar documents which are often not detected by textual based duplicate detection techniques like Shingling and Simhash. The approach utilizes hamming distance and cosine similarity (for textual and conceptual duplicate detection respectively) between two documents as their similarity measure. For performance measurement, the F-measure of the proposed approach is compared with the traditional Simhash technique. Experimental results show that our approach can outperform the traditional Simhash.

Keywords: F-measure, LSI, Shingling, Simhash, TF-IDF

1 Introduction

The near duplicates not only appear in web search but also in other contexts, such as news articles. The presence of near duplicates has a negative impact on both efficiency and effectiveness of search engines. Efficiency is adversely affected because they increase the space needed to store indexes, ultimately slowing down the access time. Effectiveness is hindered due to the retrieval of redundant documents. For designing robust and efficient information retrieval system, it is necessary to identify and eliminate duplicates.

Two documents are said to be near duplicates if they are highly similar to each other [3]. Here the notion of syntactic similarity and semantic similarity between two documents has to be carefully considered. A set of syntactically similar documents may not necessarily give positive result when tested for semantic similarity and vice versa. So two independent strategies have to be employed to detect and eliminate syntactically and semantically similar documents. Most of the traditional duplicate detection algorithms [1], [2] have considered only the aspect of syntactic similarity. Here, apart from performing textual near duplicate detection, an approach to detect

and eliminate semantically (conceptually) similar documents has been proposed. Another aspect of duplicate detection algorithms is the precision-recall trade off. Often an algorithm based on mere presence or absence of tokens in documents to be compared performs low on precision but yields a high recall considering only textually similar documents. For example if shingling [1] is implemented with *one* as the shingle size, it yields high recall but low precision. On the other hand, when techniques based on co-occurrence of words in document are used, taking into account even the order of words, its precision increases by a reasonable amount but its recall decreases. Thus, it is essential to give a right amount of importance to co-occurrence of words in documents but at the same time taking care of the recall. In the proposed approach, F-measure [9] has been used as a performance measure to give optimum precision-recall combination.

The outline of the paper is as follows: Section 2 reviews previous research work on detection of duplicate documents. Section 3 proposes the approach to effectively detect near duplicate documents. Section 4 demonstrates the experimental results which are followed by conclusion and future work covered in Section 5.

2 Related Work

One of the major difficulties for developing approach to detect near duplicate documents was the representation of documents. Broder [1] proposed an elegant solution to this by representing a document as a set of Shingles. The notion of similarity between two documents as the ratio of number of unique Shingles common to both the documents to the total number of unique Shingles in both the documents was defined as resemblance. The approach was brute force in nature and thus not practical. Charikar [2] proposed another approach which required very low storage as compared to the Shingling. The documents were represented as a short string of binaries (usually 64 bits) which is called fingerprint. Documents are then compared using this fingerprint which is calculated using hash values. Both of the above approaches became the "state of the art" approaches for detection of near duplicate documents. Henzinger [3] found that none of these approaches worked well for detecting near duplicates on the same site. Thus, they proposed a combined approach which had a high precision and recall and also overcame the shortcomings of Shingling and Simhash. Bingfeng Pi et al. [6] worked on impact of short size on duplicate detection and devised an approach using Simhash to detect near duplicates in a corpus of over 500 thousand short messages. Sun, Qin et al. [5] proposed a model for near duplicate detection which took query time into consideration. They proposed a document signature selection algorithm and claimed that it outperformed Winnowing – which is one of the widely used signature selection algorithm. Zhang et al. [7] proposed a novel approach to detect near duplicate web pages which was to work with a web crawler. It used semi-structured contents of the web pages to crawl and detect near duplicates. Figuerola et al. [8] suggested the use of fuzzy hashing to generate the fingerprints of the web documents. These fingerprints were then used to estimate the similarity between two documents. Manku et al. [4] demonstrated practicality of using Simhash to identify near duplicates in a corpus containing multi-

billion documents. They also showed that fingerprints with length 64 bits are appropriate to detect near duplicates. Martin Theobald et al. [10] took the idea of Shingling forward to include the stopwords to form short chains of adjacent content terms to create robust document signatures of the web pages.

In this paper, Shingling along with Simhash is used to detect textually similar documents. LSI [13] has also been used for detecting conceptually similar documents. Gensim [11], a Python toolkit has been used for experimental purpose and, it has been found out that the F-measure of the proposed approach is better than the traditional Simhash technique.

3 Proposed Approach

Input: A preprocessed set of documents <D_1, D_2, D_3,…, D_n>, where n is the total number of documents in the corpus.
Output: A set of documents <D_1, D_2, D_3,…, D_m> where, m<=n, which are free of near duplicates.
The proposed approach is described in Fig. 1.
1. Stop words are removed from the documents by using `preProcess()` method.
2. TF-IDF [12] is calculated for each token of each document using `getTFIDF()` method and stored in `Tfidf` array . TF-IDF weights are normalized by the length of the document so that length of the document does not have any impact on the process.
3. Each document is then represented by a set of shingles of size *three* (experimentally determined). These shingles are stored in `Shingles` array.
4. Each shingle hashed value is stored in `HashValues` array. Hashing the shingle itself (and not the token) takes into account effect of *word order* and not mere presence or absence of words as in case of traditional Simhash. Thus two documents, with same set of words used in different contexts, will not be considered near duplicates unless their word order matches to a large extent. If only single words were selected as features, two documents with largely same words, in different contexts and with different meanings are considered duplicates.
5. A 'textual fingerprint' (64 bit) `textFing` is generated for every document.

```
1. for each di in <D1, D2,…,Dn>
      preProcess(di)
      Tfidf[i] = getTFIDF(di)/len(di)
      Shingles[i] = getShingles(di, 3)//3 is shingle size
2. for each si in Shingles
      //HashValues[i] is a list of all hashes of i^th doc
      HashValues[i] = SHA1(si)//Hashing the shingles
3. textFing = Initialize2DArray(n, 64)//64 bit arrays
   for i in range(0, n)
        for each value in HashValues[i]
            for j in range(0, 64)
                  if HashValues[i][j]==1
                        textFing[i][j]+=Tfidf[i][j]
```

```
                    else
                          textFing[i][j]-=Tfidf[i][j]
   for i in range(0, n)//converting to binary
        for j in range(0, 64)
            if textFing [i][j] > 0
                  textFing[i][j] = 1
            else
                  textFing[i][j] = 0
4. for each di in <D1, D2,…, Dn>
      termDocMat = getTermDocMat(di)
      U, s, V = getSVD(termDocMat)
      lsiFing = s*(transpose(V))//lsi fingerprint
5. for i in range(0, 64)
      //rotate all textual fingerprints by 1 bit
      rotate(textFing)
      sort(textFing)
      for j in range(0, n-1)
        textSim = 64 - HamDist(textFing[j],textFing[j+1])
        concSim = cosine(jth doc, j+1th doc)
        if(textSim>th1)
            if(textSim<th2)
            //cosine similarity calculated using lsiFing
                  if(concSim>th3)
                        //add edge to Conceptual Graph
                        grConc.add(jth doc, j+1th doc)
            elif(textSim>th2 && textSim<th3)
               grText.add(jth doc, j+1th doc)
            else
               grExact.add(jth doc, j+1th doc)
6. removeDups(grText, grConc, grExact)
```

Fig. 1. The proposed approach

6. 'Conceptual fingerprints' `lsiFing` are computed for each document in the corpus. This is done using SVD (Singular Value Decomposition) matrices.
7. The list of textual fingerprints is sorted so that similar fingerprints (fingerprints which differ from each other in very less number of bits) come closer to each other.
8. The thresholds used for filtering the near duplicates are as follows:
   i. `th1` – This is a threshold that must be exceeded by textual similarity of a pair of documents to be considered for further detection. Here the idea that a pair of conceptually similar documents will also invariably exhibit some amount of textual similarity as well is used by keeping the threshold reasonably low.
   ii. `th2` - This is the textual similarity threshold. If the textual similarity of the pair of documents exceeds th2, they are considered as textual near duplicates. This is set to a quite higher value as compared to `th1` based on experimental results.

iii. `th3` - This threshold is used to detect exact duplicates. Thus it is set to a very high value.
iv. `th4` – This is the conceptual similarity threshold. If the cosine similarity [9] of two documents being checked for conceptual similarity exceeds `th4`, they are considered conceptual near duplicates.

9. Each pair of adjacent documents in the sorted list is checked for near duplicates as follows:
   i. If the documents under consideration are textual near duplicates ((64 - hamming distance) exceeds `th2`), or exact duplicates ((64 - hamming distance) exceeds `th3`), they are added to graph of Textual and Exact duplicates respectively.
   ii. If none of the conditions mentioned above are satisfied and if the (64 - hamming distance) of documents under consideration exceed `th1`, the documents are checked for conceptual similarity. If their cosine similarity exceeds a threshold, `th4`, they are added to the graph of conceptual duplicates.
10. Step (7) and (8) are repeated 64 times, each time rotating each fingerprint by one bit so as to cover all the possible near duplicate pairs. Here, the approach takes advantage of the property of the fingerprints that similar documents differ by very low number of bits in their respective fingerprints. Thus, every document need not be checked with every other document in the corpus for textual similarity.
11. Select a document from each set of duplicate documents (unique document, kept in the corpus) and remove others from input list. `removeDups` method returns the set of documents which are free of near duplicates. From each set of near duplicate documents, it keeps the longest document with a view to lose minimum information. Here, other techniques can be applied like computing each document's relevance to the query and outputting the one with the highest value.

4 Experimental Results

For experimental purpose, *20 Newsgroup Dataset* [14] is used. The *20 Newsgroup Dataset* is a collection of approximately 20,000 newsgroup documents, partitioned (nearly evenly) across 20 different newsgroups. The *20 Newsgroup* collections are widely used for experiments in areas of text processing and clustering. It offers a group of documents which already contains some duplicates, making it suitable for testing near duplicate detection algorithms. The proposed duplicate detection approach was implemented using Python and two kinds of experiments were majorly performed on dataset as described below.

4.1 Determination of optimum values for parameters involved in algorithm

For figuring out the optimal combination of parameters, algorithm was run for a range of different thresholds and Shingle sizes. The thresholds `th1`, `th3` and `th4` are fixed to a reasonable value. As `th1` is the minimum similarity threshold which two documents must pass to be considered for duplicate detection, it can be set to an

intuitively low value. Similar argument holds true for `th3` and `th4`. So, only the Shingle size and `th2` need to be determined experimentally. Fig. 2 described the precision, recall and F-measure with respect to Shingle size (where threshold is set to optimum). As the graph clearly shows, the Shingle size = 3 gives the best results with highest F-measure of 0.88. When Shingle size is very low, the algorithm tends to neglect the effect of word order while testing two documents for duplicates. On the other hand, when Shingle size is very high, algorithm can only find duplicates which are exact copies of each other and lacks ability to find *near duplicates*. A balance

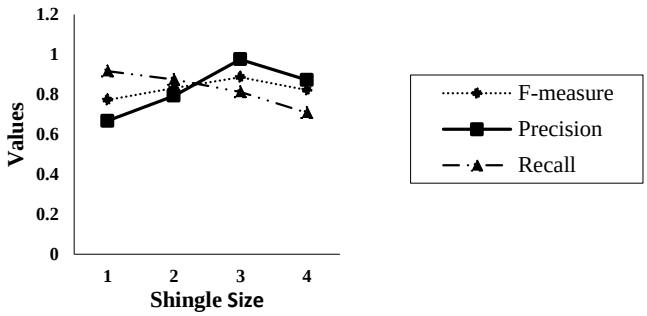

Fig. 2. F-measure, Precision and Recall v/s Shingle size

between these two extremes is obtained using a sequence of tokens of length 3 each as suggested by the results. Fig. 3 shows the graphs of precision, recall and F-measure with respect the value of `th2`. Here, it can be seen that as the value of `th2` increases, precision increases and recall decreases. When th2 value is increased, two documents are considered to be near duplicates only when they have a large chunk of text in common and thus this does not cover all near duplicate documents, leading to low recall. Whereas, when `th2` value is very low, almost every document is considered as a near duplicate covering all the actual near duplicate documents but performing poorly on precision. F-measure is used to determine the optimum `th2` value which turns out to be 46.

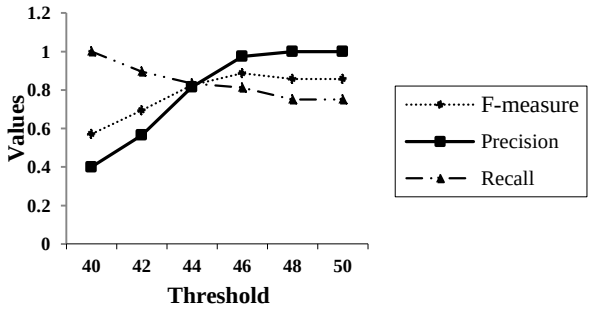

Fig. 3. Threshold (th2) v/s F-measure, Precision and Recall

4.2 Comparison of the proposed approach with traditional Simhash implementation

Simhash-0.1.0, a Python library is used for comparing the proposed approach with Simhash. To demonstrate the work and for comparison purposes, a corpus of 128 documents from *20 Newsgroup* dataset out of which 48 documents were near duplicates (as mentioned in [14]) has been considered. Both the algorithms were run for same set of conditions (the corpus and the thresholds). The results are shown in Fig. 4. Simhash could only identify 7 documents as near duplicates whereas the proposed approach could identify 40 near duplicate documents. Simhash performs excellently on precision but poorly for recall. This is because, Simhash, when comparing two documents, totally relies on syntactic occurrences of words in the documents and does not take into account the context of occurrence of words. The use of Shingling along with Simhash and LSI, in the proposed approach, takes into account not only the syntactic occurrence of words in the documents, but also the context or the co-occurrence of terms. This leads to not only high precision but high recall as well, because both – syntactic and semantic near duplicates are identified.

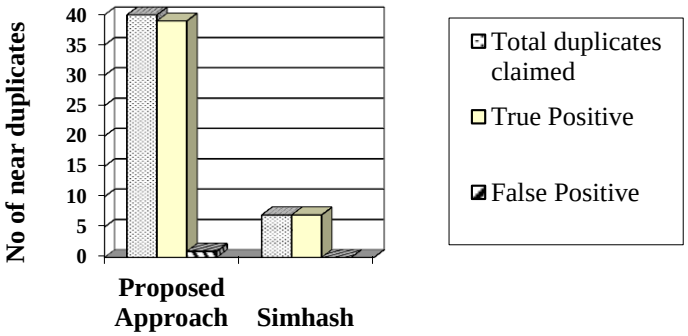

Fig. 4. Comparison of Proposed Approach and Simhash based on the No. of duplicates detected

5 Conclusion and Future Work

In this paper, the proposed approach uses a combination of Shingling and Simhash techniques to detect textual near duplicates. For detecting conceptual near duplicates, LSI is used to generate fingerprints of the documents. Optimal Shingle size and threshold is determined by experiment using F-measure as the performance measure. Experimental result show that although the algorithm gives less precision (0.975) in comparison to Simhash (1.0) but it has a very high recall yielding a good F-measure (0.886) as compared to Simhash (0.254). With the help of LSI technique, the proposed approach detects even the conceptual duplicates which are often missed out by traditional Simhash. In addition, use of Shingling makes the approach more robust in terms of its precision. As part of future work, near duplicate document detection

algorithm can be designed which is less susceptible to the length of the document. Also, since the algorithm has to process a large corpus of documents, efforts can be made to implement such algorithm on a distributed computing framework such as Map Reduce using Hadoop. This will greatly increase the efficiency of the algorithm.

References

1. Broder.: Identifying and filtering near-duplicate documents. In COM '00: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, pp. 1-10. Springer-Verlag, London, UK, (2000).
2. Charikar.: Similarity estimation techniques from rounding algorithms. In STOC'02: Proceedings of the 34th Annual ACM symposium on Theory of computing, pp. 380-388. ACM, New York, NY, USA (2002).
3. Henzinger.: Finding near-duplicate web pages: a large-scale evaluation of algorithms. In SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 284-291. ACM, New York, NY, USA (2006).
4. Gurmeet Singh Manku, Arvind Jain, Anish Das Sharma.: Detecting Near-duplicates for web crawling. In WWW / Track: Data Mining (2007).
5. Sun, Qin, Wang.: Near Duplicate Text Detection Using Frequency-Biased Signatures. In Web Information Systems Engineering – WISE, Lecture Notes in Computer Science Volume 8180, pp. 277-291. (2013).
6. Pi, Fu, Wang, Han.: SimHash-based Effective and Efficient Detecting of Near-Duplicate Short Messages. In Proceedings of the 2nd Symposium International Computer Science and Computational Technology.
7. Y.H. Zhang, F. Zhang.: Research on New Algorithm of Topic-Oriented Crawler and Duplicated Web Pages Detection. In Intelligent Computing Theories and Applications 8th International Conference, ICIC, Huangshan, China, July 25-29. (2012).
8. Figuerola, Díaz, Berrocal, Rodríguez.: Web Document Duplicate Detection using Fuzzy Hashing. In Trends in Practical Applications of Agents and Multiagent Systems, 9th International Conference on Practical Applications of Agents and Multiagent Systems, Volume 90, pp. 117-125 (2011).
9. Pang-Ning Tan, Vipin Kumar, Michael Steinbach, Introduction to Data Mining, Pearson.
10. Theobald, Siddharth, Paepcke.: SpotSigs: Robust and Efficient Near Duplicate Detection. In Large Web Collections in SIGIR'08 July 20–24 (2008).
11. Rehurek, Radim a Petr SOJKA.: Software Framework for Topic Modeling with Large Corpora. In Proceedings of LREC workshop New Challenges for NLP Frameworks. Valleta, Malta: University of Malta, 2010. s. 46—50, 5 s. ISBN 2-9517408-6-7 (2010).
12. Robertson.: Understanding Inverse Document Frequency: On theoretical arguments for IDF, in Journal of Documentation 60 no. 5, pp. 503–520.
13. Golub, Reinsch.: Singular value decomposition and least square solutions in Numerische Mathematik 10. IV. Volume 14, Issue 5, pp. 403-420 (1970).
14. Celikik, Bast.: Fast error-tolerant search on very large texts in SAC '09 Proceedings of the ACM symposium on Applied Computing pp. 1724-1731 (2009).