Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 999 – 1013
© TÜBİTAK
doi:10.3906/elk-1906-30

A fast text similarity measure for large document collections using multireference cosine and genetic algorithm

Hamid MOHAMMADI*, **Seyed Hossein KHASTEH**
Department of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran

Received: 06.06.2019 • **Accepted/Published Online:** 27.10.2019 • **Final Version:** 28.03.2020

Abstract: One of the critical factors that make a search engine fast and accurate is a concise and duplicate free index. In order to remove duplicate and near-duplicate (DND) documents from the index, a search engine needs a swift and reliable DND text document detection system. Traditional approaches to this problem, such as brute force comparisons or simple hash-based algorithms, are not suitable as they are not scalable and are not capable of detecting near-duplicate documents effectively. In this paper, a new signature-based approach to text similarity detection is introduced, which is fast, scalable, and reliable and needs less storage space. The proposed method is examined on standard text document datasets such as CiteseerX, Enron, Gold Set of Near-duplicate News Articles, and other similar datasets. The results are promising and comparable with the best cutting-edge algorithms considering accuracy and performance. The proposed method is based on the idea of using reference texts to generate signatures for text documents. The novelty of this paper is the use of genetic algorithms to generate better reference texts.

Key words: Text similarity, near-duplicate, reference text, genetic algorithm

1. Introduction

The main task of a search engine is searching. Therefore, it has to acquire a set of web pages to search through, which is called index. A search engine works fast and reliably when its index is as concise as possible without missing any possible web pages. For this purpose, the DND documents must be removed from the index. Another component of a typical search engine that is prone to DND text documents problems is a crawler. The crawler is a component in a search engine that has the responsibility of surfing the web and downloading web pages to index. A crawler faces the problem of DND web pages since a significant number of web pages on the web are DND web pages [1–3]. In order to have a formal definition of DND web pages, we can say that duplicate web pages are the pages that are identical in terms of content, but they are accessible using multiple URLs. On the other hand, near-duplicate pages are the web pages with slight differences, such as changed date or some other minor edits, but they are not the same [4, 5]. One of the problems with DND web pages is that they increase the size of the search index. Such an index could reduce the quality of search engine results and increase the computational power needed to perform all kinds of tasks on them [2]. An index works better when the search engine is able to detect the duplication or near-duplication of web pages in a speedy and accurate manner [2, 3, 6]. The common approaches to overcome this problem include brute force approaches and hash-based algorithms. Approaches using brute force techniques compare each document with all other documents using a similarity measure like cosine [1] or Jaccard [7]. These approaches are not

*Correspondence: mohammadi2823@email.kntu.ac.ir

suitable for large-scale document collections containing hundreds of thousands of text documents such as the World Wide Web. These approaches are not efficient since all documents must be compared one by one. The time complexity of comparing all documents is $O\left(n^2\right) * L$, where L is the mean length of documents. One approach to solve this problem is to reduce L using hashing methods. These methods generate a signature (which is much smaller than the document itself) for each document. For example, a 128- or 160-bit length hash is created for each document using MD5 or SHA1 algorithms. As a result, it is possible to compare the signature of documents instead of comparing the documents and reduce the processing time. In these methods, the required computational power is decreased drastically. The most significant shortcoming of such methods is that most of these approaches are incapable of detecting near-duplication as the amount of similarity between two documents cannot be measured using hashes generated by these hashing methods [4].

In this study, a new approach to overcome these odds is proposed. The new approach is based on the multireference cosine text similarity algorithm [8]. The new approach is scalable, reliable, and fast. The proposed method can also be used in some other applications such as document clustering, plagiarism detection, and recommender systems. The main contribution of the proposed method is the achievement of the highest recall in near-duplicate detection tasks among state of the art approaches to near-duplicate document detection algorithms in large document collections.

The rest of this paper is structured as follows: In Section 2, some of the related works are reviewed. In Section 3 the new approach is explained. Section 4 will show the experimental results. Finally, Section 5 presents conclusions and future works.

2. Related works

Search engines are the gateway to the web [9]. Indexes are one of the key components of a search engine and have a deep influence on its performance [10]. One of the properties of a good index is that it is not suffering from DND entries.

Several studies have been conducted in order to solve the problem of finding and removing DND documents and web pages. These studies could be classified into two major classes: URL-based methods and content-based methods. The URL-based methods find common patterns in the URLs of web pages and recognize DND web pages using these patterns [11]. One of the first works on this branch of approaches was done by Bar-Yossef et al. [12]. Later Dasgupta et al. [13] and Koppula et al. [11] extended their works on detecting DND web pages using URLs. Koppula et al. analyzed web server logs to find patterns in URLs that point to DND web pages.

The other class of DND text detection methods is the content-based methods class. The new content-based attempts to solve the problem of DND text document detection were brute force approaches. In these approaches, a server would compare each document with all other documents using a similarity measure such as cosine text similarity [14]. Brute force methods are accurate because they compare all the documents one by one. However, the disadvantage of these approaches is their performance. Therefore, these approaches are not suitable for applications with a great number of text documents. Later, researchers focused on finding DND documents with a lesser amount of computational power and storage space. The cornerstone of these studies was the works of Manber [15] and Heintze [16], which were focused on the adjacent characters' resemblance. Meanwhile, Brin [17] came up with a system that used hashes to detect copyright violations. Later, Broder et al. [1] introduced the shingling algorithm, which was used in the AltaVista search engine as a duplicate document detection algorithm. Border's algorithm does not use any prior knowledge of the target language, which makes it more suitable for the multilingual ecosystem of the web. Border's method uses the Jaccard similarity measure to compare documents shingles. Next year, locality sensitive hashing or LSH, which is an approximate approach to find DND documents, was introduced by Indyk et al. [18]. The approximate approach to DND text detection ensures the performance of the algorithm but increases the number of false-positive and false-negative decisions [19]. I-match [20] is another innovation to find DND text documents, which uses lexical methods. I-match uses a lexicon created from a large text corpus and creates a signature for each document, using the SHA1 hashing algorithm. The similarity of the signatures generated by the I-match algorithm will show the probability of duplication. Later, Sarawagi et al. [21] used the inverted-index method, which is only able to find duplicate documents.

One of the most efficient and frequently used algorithms for DND document detection was introduced by Charikar [22]. Charikar's algorithm, Simhash, uses dimensionality reduction techniques to generate fixed-length hashes for each document. The Simhash algorithm has shown promising results [23] and is being used by many search engines around the world, such as Google [4]. Henzinger [3] combined Broder's and Charikar's algorithms to achieve higher precisions.

Another state of the art algorithm is SpotSigs [7]. SpotSigs showed that some parts of documents have a higher impact on the measured similarity of the documents compared to the other parts. Hajishirzi et al. [24] introduced a domain-specific algorithm that uses a real-valued k-gram summary vector as a signature for each document and can be adapted to use different similarity measures like cosine and Jaccard to detect duplicate and near-duplicate text documents.

As the amount of available data and the number of web pages are increasing rapidly in recent years, the use of big data (map-reduce) techniques is becoming common among studies for DND document detection. Lin [25] and Vernica's [26] works are of this type, and they tried to adapt current methods to big data applications and frameworks.

Some of the recent studies are focused on DND document detection systems or new hybrid approaches, which combine the previous algorithms and enhance the precision or performance. For example, the works of Pamulaparty et al. [27] can be mentioned. In that research, a new architecture for DND document detection was introduced. The system introduced by Pamulaparty et al. first parses a web page and extracts its texts from it. Then, after stop word removal and stemming, it adds the new document to its database and checks the database for documents with the highest common words and marks them as DND texts. Varol et al. [28] introduced a new hybrid approach to DND document detection. They combined shingling with Jaro distance and word usage frequency to enhance the shingling algorithm [28].

Zhang et al. [29] introduced one of the latest and most efficient methods to find DND text documents. They used the idea of normalized compression distance, or NCD [30], and combined it with the idea of hashing in order to solve the problem of NCD with medium and large files. NCD is a similarity metric that is universal and parameter-free. NCD is based on the Kolmogorov complexity [31]. The main idea of NCD is that if two text documents have more common information, the result of compressing these two files together is smaller. In other words, the shorter the compressed form of two text documents is, the more similar those two texts are. The Zhang method, which is called SigNCD, is both accurate and high-performing. Zhang also introduced another algorithm called SpotSigNCD [29]. SpotSigNCD is based on SigNCD but uses the stop word-spot signature extraction method instead of the original signature extraction method of SpotSig [29]. SpotSigNCD has better precision than SigNCD, but its recall is not as good as that of SigNCD [29]. In this paper, a new signature-based method is introduced, which has competitive accuracy and performance.

3. Proposed algorithm

The proposed method works based on an idea called the multireference cosine text similarity algorithm [8]. Each text (reference text and text document) is considered as a sequence of 3-grams. In this algorithm, in order to generate the signature of the text document (D_i), it compares D_i with different parts of the reference text using the cosine text similarity measure. The cosine text similarity algorithm is chosen as our core similarity measurement algorithm as the cosine text similarity algorithm is more accurate than other similar algorithms such as the Pearson correlation coefficient (PCC), Jaccard, and mean square difference (MSD) in terms of measuring the similarity or difference between texts [32]. The result is a decimal number for each comparison. The algorithm puts these decimal numbers all together to create a vector and considers this vector as the signature of D_i. The algorithm uses the signature to calculate the similarity between different text documents. The generated signature has the property that if the original text documents are similar, the signatures of those text documents are similar as well. Therefore, the algorithm can detect DND text documents using this signature. The proposed method could be used for measuring the similarity degree between documents in high volume text document datasets in a fast and efficient way. The overall structure of a system, using the proposed algorithm to find DND documents, is shown in Figure 1.

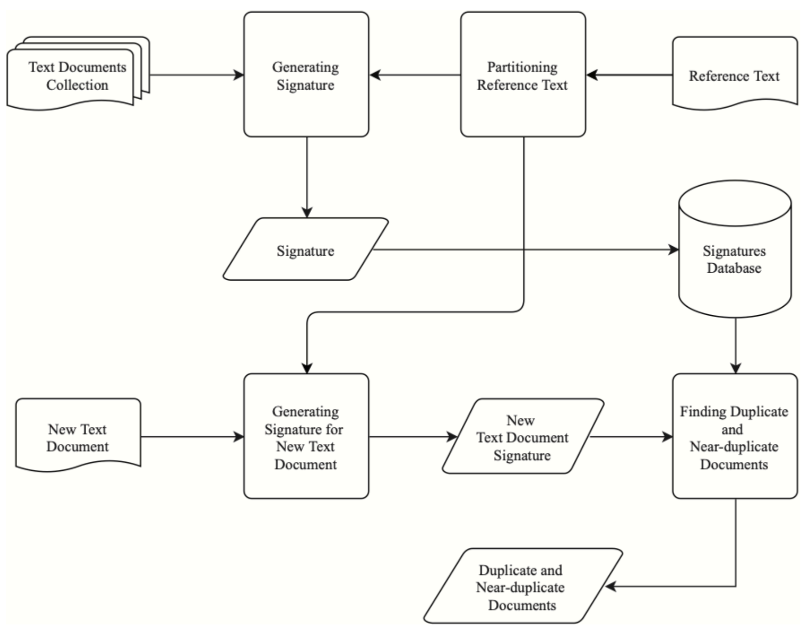

Figure 1. Structure of DND documents detection system using multireference cosine.

The system in Figure 1 consists of text documents, reference text, document-reference comparison, signatures database, and similarity measurement. The system components are explained as follows:

3.1. Text documents

The text documents could be text files, web pages, or any other type of files which consist of text. Text documents can be seen in two parts of Figure 1:

"Text Documents Collection": The Text Documents Collection is a database which contains all previously collected text documents.

"New Text Document": The new text document is a text document for which the system wants to find its DND text document in the database.

3.2. Reference text

The reference text is one of the main components of the multireference cosine algorithm. The reference text is a sequence of 3-grams, which is used by the algorithm to generate a signature for each text document. N-grams are the essential elements of comparison in the cosine text similarity algorithm, which is the core of the multireference cosine algorithm. Longer N-grams can be more effective in terms of measuring the similarity or difference between two texts. On the other hand, a greater value of N means a higher number of possible permutations for N-grams, which makes the reference text search space larger, hence increasing the processing power and time required to find a desirable reference text. Considering this trade-off between shorter and longer N-grams, 3-grams are found optimum in terms of effectiveness in measuring the similarity between documents and performance [8]. Reference text could be generated with the help of methods such as the information gain method. In this paper, a new method of generating the reference text is proposed. The new method uses genetic algorithms to find better reference text to achieve more accuracy and better performance for the multireference cosine text similarity algorithm. By using this new method of reference text generation, the performance of the whole system outperforms some of the best state of the art algorithms like Simhash [22], which Google reported using as a DND web page detection algorithm in its search engine [4]. Using this new method, the previous method of generating reference text that uses information gain theory to minimize the mean absolute error of the multireference cosine algorithm [8] is outperformed, too.

3.3. Reference text partitioning

To generate a signature, the multireference cosine text similarity algorithm splits the reference text into several parts. It creates multiple reference texts from the original reference text. The multireference cosine algorithm compares a document with the reference text parts using the cosine text similarity measure. The result of each comparison is a decimal value between zero and one. The greater the number is, the more similar the text document is to that part of the reference text. In other words, the algorithm examines how frequent the 3-grams of that specific reference text are in the selected text document. The best 3-grams to be a part of the reference text are the 3-grams that have the highest information about how similar or different two documents are. Therefore, the existence or absence of each 3-gram inside the reference text may have a considerable impact on the signature of a document. Given this fact, if the algorithm compares a document with one reference text that contains many 3-grams, it has missed some of the information that each 3-gram could have given us. For example, there may be a reference text that contains $(3 - gram_n)$ and $(3 - gram_m)$.

On the other hand, there is a document (D_i) that contains $(3 - gram_n)$ and has high similarity with the part of reference text containing $(3 - gram_n)$ and zero similarity to the other parts of reference text. There is also another document (D_j) that contains $(3 - gram_m)$ and has high similarity with the part of reference text containing $(3 - gram_m)$ and no similarity with the other parts. These two documents get the same signatures because they contain the same number of common 3-grams with the reference text, and they will be detected as similar. If the algorithm divides the reference text into several parts, each containing a limited number of 3-grams, it can harness more information about what each 3-gram can tell us about a document. If the reference text contains all possible 3-grams in the character set that the documents are made of, by reducing the size of each partition of reference text to a single 3-gram, the algorithm is creating the vector space model of the document, which is somehow a complete numeric representation of each document. In the scenario above, having the exact vector space model of each document will guarantee the cosine text similarity accuracy. However, its performance is similar to a naive brute force text comparison algorithm, which is not desirable for large text document collections.

Nevertheless, all the 3-grams are not equally important regarding similarity detection, and some 3-grams contain more information than others. By selecting these important 3-grams and removing other 3-grams from reference text, the multireference cosine text similarity algorithm can reduce the reference text's size drastically and increase the performance. The exact number of partitions depends on the desired accuracy and performance. The effect of the number of partitions on the final accuracy and performance will be discussed in Section 4.

3.4. Signature generation

As mentioned before, in order to generate a signature for each text document, the algorithm compares the text document with each reference text part using cosine text similarity measure. The result of each comparison is a decimal value between zero and one. The algorithm stores the comparison results in an array. It uses this array as the signature of the text document.

3.5. Finding DND documents

Since the signature is an array of decimal values, it can be considered as a vector. In order to compare documents in this system, the algorithm can compare their signatures, which are stored in a database, using a vector similarity measure such as cosine text similarity. To find DND text documents, the algorithm uses two predefined thresholds, t_1 and t_2, that can take values between 0 and 1. These threshold values are determined empirically and using trial and error for each specific application. Using these thresholds, the algorithm can mark a document with a high degree of similarity s (s $\geq t_1$) as a candidate for a duplicate document or as a candidate for a near-duplicate document if $t_1 \geq$ s $\geq t_2$. A multireference cosine algorithm with greater values for t_1 and t_2 finds documents with a higher degree of similarity to the original document as DND documents, in comparison to the same algorithm with lower values for t_1 and t_2.

3.6. Generating reference text

The reference text has a significant impact on the performance of the newly proposed method. Therefore, choosing an excellent reference text is one of the most critical parts of the multireference cosine algorithm. Before this study [8], the idea of information gain was used to generate reference texts. The results were acceptable, yet a better reference text is needed to acquire better results. In this study, a genetic algorithm is used in order to generate a better reference text. The reference text is a sequence of characters so that it can be considered as a chromosome, and accordingly, it is possible to use genetic algorithms to find good reference texts. The flowchart of the used genetic algorithm is presented in Figure 2. The components of such a system are explained as follows:

3.6.1. Initial population

The initial population can be generated using different methods, but in order to increase the quality of the final results and reduce the number of generations needed to reach a desirable result, N-grams with high information gain to generate the reference texts are used. An N-gram is an N-character-long string. In this implementation of the cosine text similarity algorithm, a 3-gram model is used in order to compare two texts. A total of 9000

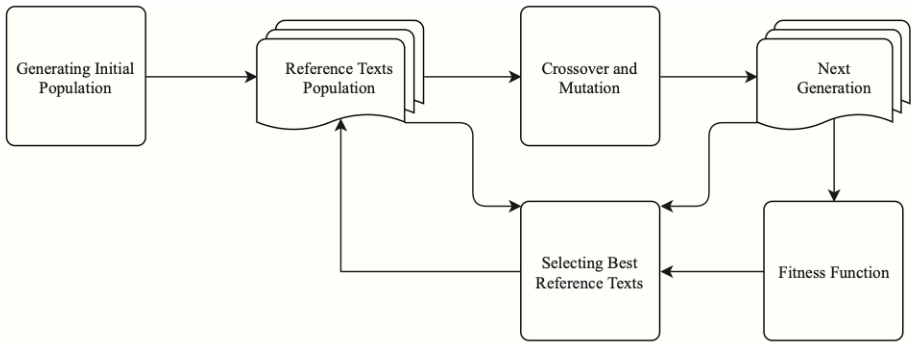

Figure 2. Generating reference texts using genetic algorithm.

3-grams with highest Tf-idf score to create the initial population of reference texts are extracted from the Open American National Corpus [33]. To score a document using the Tf-idf method, the term (in this case, 3-gram) frequency in each document and the inverted document frequency (the frequency of documents in which the term appeared) are used. By using the Tf-idf scoring method, we ensure that the selected 3-grams are the 3-grams with the highest information gain, which are the most useful 3-grams to determine how much two text documents are similar or different. After selecting 3-grams, the initial population is generated by randomly combining some of the highly scored 3-grams to a fixed length.

The final accuracy and performance of the multireference cosine text similarity algorithm directly depend on the reference text length. If the algorithm uses a concise reference text, although the reference text is made by using the best possible 3-grams, the algorithm cannot exceed an accuracy limit, because of the limited number of possible 3-grams in a short reference text.

On the other hand, if the reference text contains all possible 3-grams in a language or character set, the increase in the reference text length will no longer increase the final accuracy. Besides, the longer reference text makes the multireference cosine text similarity algorithm slower. Therefore, in order to achieve good overall accuracy and performance, we should consider choosing an optimized reference text length. In general, there is a trade-off between accuracy and performance of the multireference cosine text similarity algorithm. The effect of different reference text lengths on the final accuracy will be discussed in Section 4.

3.6.2. Next generation

The genetic algorithm uses methods such as crossover and mutation to generate the intermediate population from the current population. Crossover and mutation can be done in different ways and with different rates. As a result of conducted tests, the best methods and rates that will help to reach the best results in the case of optimizing reference texts are found. The method that is used for the crossover is single cut crossover, which means that the algorithm cuts two reference texts in a random length and displaces the cuts. The method used for mutation is to randomly replace 10 percent of 3-grams of a reference text. As we mentioned earlier, 3-grams with higher Tf-idf scores make reference texts more effective in terms of accurately measuring the similarity or dissimilarity between two documents; accordingly, the selected 10 percent of 3-grams are replaced with randomly chosen 3-grams from a list of 3-grams with high Tf-Idf scores in the OANC corpus.

3.6.3. Fitness function

In this step, as a fitness function, reference texts get scores according to how accurately the multireference cosine algorithm can measure the similarity degree between different documents. As mentioned in Section 3, the higher accuracy of the cosine text similarity algorithm compared to other text similarity measures such as PCC, Jaccard, and MSD in terms of calculating the degree of similarity between two texts makes it a gold standard for text similarity [32]. A version of the multireference cosine text similarity algorithm is implemented using the generated reference text over a number of randomly selected documents in a dataset, comparing them one by one. Similar to any form of statistical modeling, the randomness of sampling can affect the outcome of an algorithm, but large samples from the original data could provide a precise model. On the other hand, larger samples demand more processing power to process. In this study, sample sizes, which are mentioned for each dataset in Section 4.1, are determined to obtain desired accuracies with the least processing power required. The final score is the mean absolute error of the results of the multireference cosine algorithm in comparison with the results of a traditional cosine text similarity algorithm. The fitness function is executed for each reference text in the intermediate population. The final score is shown in Eq. 1:

$$Final\ Score = \frac{\sum_{i=0}^{N-1}\sum_{j=i+1}^{N}|Multi\,reference\,cosine(D_i - D_j) - cosine(D_i - D_j)|}{\frac{N(N-1)}{2}}, \tag{1}$$

where N is the number of randomly selected documents, D_i and D_j are ith and jth documents in the randomly selected document set, and $\frac{N(N-1)}{2}$ is the total number of comparisons.

3.6.4. Selecting reference texts

Next, the next generation is selected from the intermediate population. The selection criterion is the score that each reference text acquires through the fitness function. The reference texts with higher scores are selected. The number of selected reference texts is the same as the number of reference texts in the main population. This new population takes the place of the previous main population.

The process of generating the intermediate population, scoring intermediate population reference texts, and selecting the next generation reference texts continues until the accuracy of the whole system using the latest generation of reference texts reaches a predefined threshold. Usually, the average number of needed generations to reach the desired threshold is about 50 generations.

4. Test results

In order to test the newly generated reference texts, the whole system must be tested using the new reference text. The multireference cosine is tested using generated reference texts on several datasets. Reported run times are computed using a personal computer running Ubuntu 17.04 and OpenJDK Java 1.8.0, with a core i7-4510U CPU (all cores are used) and 8 gigabytes of RAM. Datasets and test results are as follows:

4.1. Datasets

Several standard datasets that have been used in similar studies are gathered. The datasets are:

4.1.1. Citeseerx dataset

Citeseerx is a scientific literature digital library and search engine that is focused on scientific literature in the field of computer and information science. The Citeseerx dataset[1] contains more than 6 million scientific papers and their metadata. Ten thousand randomly selected documents in the Citeseerx dataset have been used for the test purposes.

4.1.2. TREC 2005

TREC stands for Text Retrieval Conference. It is a conference that is held annually, and it is focused on encouraging studies in the field of information retrieval. The TREC 2005 public spam corpus[2] contains 92,189 email messages. Each message is stored in a text file, so it can be used for evaluating the DND text document detection method. The portion of the dataset that is used for test purposes is consist of ten thousand randomly selected email messages.

4.1.3. DMOZ

DMOZ is an open-content web directory of World Wide Web URLs, which is also known as an "open directory" project. The original DMOZ dataset[3] contains over two million records. About ten thousand URLs are randomly selected, and the corresponding web pages are downloaded. All HTML tags are removed from the downloaded web pages, and the content of each web page is extracted. The final documents are used for the tests.

4.1.4. Newsgroups

The 20 Newsgroups dataset[4] is a collection of about 20,000 newsgroup documents. Ken Lang originally collected this dataset for his "Newsweeder: Learning to Filter Netnews" paper [34]. The data are categorized into 20 topics. Just as for other datasets, ten thousand documents in this dataset are randomly chosen and used for test purposes.

4.1.5. OpenDNS public domains

The OpenDNS public domains dataset[5] was originally a list of several randomly selected domains stored on OpenDNS domain name servers. This dataset contains a list of ten thousand domain names. The corresponding web pages of these domain names are downloaded, and all HTML tags are removed from them. The remaining content is used for test purposes.

4.1.6. Enron

Enron was one of the biggest companies in the United States in the 1990s. Later, in 2001, the company declared bankruptcy. The company faced a financial crisis because of creative and systematic accounting fraud. The Enron dataset[6] is a large collection of emails from or to Enron employees. It contains over one hundred thousand files. About ten thousand randomly selected files in this dataset are used for test purposes.

[1] CiteSeerX (2017). CiteSeerX data [online]. http://csxstatic.ist.psu.edu/downloads/data [accessed 5 May 2017].

[2] TREC (2005). 2005 TREC Public Spam Corpus [online]. https://plg.uwaterloo.ca/~gvcormac/treccorpus/ [accessed 5 May 2017].

[3] Dmoz (2016). Parsed DMOZ data [online]. https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/OMV93V [accessed 5 May 2017].

[4] Ken lang (1995). 20 Newsgroups [online]. http://qwone.com/~jason/20Newsgroups/ [accessed 26 October 2016].

[5] OpenDNS (2014). OpenDNS Top Domains List [online]. https://github.com/opendns/public-domain-lists [accessed 5 May 2017].

[6] Enron (2015). Enron Email Dataset [online]. https://www.cs.cmu.edu/~enron/ [accessed 7 January 2017].

4.1.7. Gold set of near-duplicate news articles

The gold set of near-duplicate news articles[7] is a dataset consisting of over 2,000 news articles clustered into 68 categories. This dataset was originally a part of the Stanford Web Base crawl [35]. It is a great dataset to evaluate a near-duplicate detection algorithm because human assessors have manually marked the near-duplicate documents in this dataset. Also, it was used by many researchers studying detection of DND text documents in large collections to evaluate their algorithms [7, 24, 29]. Therefore, in order to compare the new method with other algorithms, the new approach is examined in this dataset.

4.2. Tests

To test the new method, first, the desired reference text is generated using the genetic algorithm. The procedure explained in Section 3.2 is used in order to generate the reference text. To find the best reference text, the cross-validation technique is used. The considered dataset was divided into two parts, one containing 80 percent and the other one 20 percent of the dataset's documents. The first part (80 percent) of the dataset is used in order to evaluate each reference text in a generation using the fitness function. After completing the genetic algorithm process, the second part of the dataset (20 percent) is used in order to evaluate the final reference text. Due to the random nature of the genetic algorithm, this process is executed ten times for each test in order to average out the results and avoid very good or very bad random reference texts. Finally, the reference text with the highest fitness on the second part of the dataset (20 percent) is chosen as the winner. Then the winner reference text and multireference cosine text similarity algorithm are used for detecting the DND documents in the other datasets. The mean absolute error of the algorithm is calculated using Eq. 1 for each test. For DND text document detection, precision is the ratio of the number of correctly predicted DND text documents to the whole number of documents that are predicted as DND. On the other hand, recall is the ratio of the number of correctly predicted DND text documents to the whole number of DND text documents. The F1-score is a weighted average of precision and recall. The equations to calculate the precision, recall, and F1-score are as shown below (Eq. 2 to 4):

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3}$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

Here, a true-positive is a document that is DND, and it is correctly predicted as DND by the system. False-positive is a document that initially is not a DND document, but the system has detected it as a DND document. Moreover, false-negative is a document that is originally a DND document, but the system incorrectly predicted it as a non-DND document. Fifty generations are used in order to find the best reference text because there is no significant change in the fitness of the best reference texts after 50 generations. In order to find a suitable population size, an experiment was performed. The genetic algorithm is executed to find the optimal reference text on the Citeseerx dataset with different population sizes. The test results are shown in Table 1.

[7] GoldSet (2008). Gold set of near-duplicate news articles [online]. http://adrem.ua.ac.be/~tmartin/ [accessed 7 January 2017].

Table 1. Comparing performance of proposed method with different population sizes.

Train dataset	Population size	Mean absolute error	Runtime of each iteration (seconds)
Citeseerx	50	0.1083	9
Citeseerx	100	0.1053	20
Citeseerx	150	0.1041	30

Table 1 shows that by increasing the population size, the final accuracy of the algorithm increases. However, this improvement is not valuable since the runtime of the genetic algorithm has a linear relation with the population size. In order to have desired results and also reduce the runtime of the genetic algorithm, the population size of 100 chromosomes is used for the tests.

In Table 1, the results of generating the reference text using each of the datasets are presented. As mentioned above, 80 percent of documents of each dataset were used to evaluate the fitness of reference texts in the genetic algorithm. The remaining 20 percent of the dataset is used in order to evaluate the final reference text. In each of the following tests, reference texts with the length of 1000 3-grams were generated. The number of reference text partitions used in each test is 150. A reference text with 1000 3-grams length and 150 parts is used with the intention of enhancing algorithm results by using these values. The cross-validation process is executed on the datasets. The final evaluation results are shown in Table 2.

Table 2. Cross-validation evaluation results.

Train dataset	Test dataset	Mean absolute error
CiteseerX	CiteseerX	0.1047
TREC 05	TREC 05	0.1246
DMOZ	DMOZ	0.1292
20 Newsgroups	20 Newsgroups	0.1281
OpenDNS public domains	OpenDNS public domains	0.1070
Enron	Enron	0.1164

As shown in Table 2 the new approach has an average of 0.1183 difference in term of mean absolute error from the cosine text similarity algorithm. It can be concluded that the multireference cosine text similarity algorithm is accurate and reliable.

In another test, the reference text is generated using the Citeseerx dataset as the training dataset, and it is evaluated on the other datasets to test the capability of the new method to be generalized to other datasets. As was mentioned before, the algorithm achieves better results with a reference text with the length of 1000 3-grams and 150 parts. Therefore, this configuration can be used for this test, too. The test results are shown in Table 3.

As shown in Table 3, the test results on the other datasets are a little worse but generally are promising. In order to understand the effects of reference text length and the number of partitions on the accuracy of the final results, reference texts with different lengths and also using different number of partitions are generated. The Citeseerx dataset is used for these tests. The results are shown in Table 4.

Table 4 shows that, in general, increasing the length of reference text increases the accuracy of the multireference cosine text similarity algorithm. This was discussed in Section 3.6.1. Furthermore, the results show that the relation between the number of partitions and the final accuracy is somehow ambiguous. Nevertheless,

Table 3. Training and evaluating over different datasets.

Reference text dataset	Evaluation dataset	Mean absolute error
Citeseerx	TREC 05	0.1444
Citeseerx	DMOZ	0.1355
Citeseerx	20 Newsgroups	0.1012
Citeseerx	OpenDNS public domains	0.1355
Citeseerx	Enron	0.1389

Table 4. Test results using different reference text lengths and different numbers of partitions.

Reference text length	Number of partitions	Mean absolute error
450 3-grams	100	0.1226
450 3-grams	150	0.1240
450 3-grams	300	0.1281
600 3-grams	100	0.1190
600 3-grams	150	0.1186
600 3-grams	300	0.1197
1000 3-grams	100	0.1193
1000 3-grams	150	0.1047
1000 3-grams	300	0.1135

it seems that for different lengths of the reference text, dividing the reference text into 150 parts leads to better results. Table 4 shows that very long or very short reference partitions decrease the algorithm's accuracy.
The new approach is tested on the gold set of near-duplicate news articles in order to compare the new algorithm with the other recent approaches. The precision and recall of other approaches are collected from a paper by Zhang et al. [29]. For this test, a reference text is used, generated using the Citeseerx dataset, with length of 1000 and 150 parts. The precision, recall, and F1-score of the proposed algorithm and other algorithm are shown in Table 5.

Table 5. Comparing proposed method with other approaches.

Algorithm	Precision	Recall	F1-score	Runtime (ms)
SigNCD	0.95	0.89	0.92	7838
SpotSigNCD	0.97	0.87	0.92	10853
Proposed method	0.87	0.98	0.92	10950
NCD	0.90	0.78	0.83	53829
SpotSigs	0.90	0.77	0.83	12824
Simhash	0.85	0.45	0.59	13010

As presented in Table 5, the multireference cosine achieved an F1-score equivalent to the state of the art algorithms. In addition to that, the proposed algorithm obtained the highest recall between the algorithms presented in Table 5, which gives it special confidence in detecting the highest number of DND documents in a target document collection. The runtime measured for the proposed algorithm is the time required for its comparisons and does not include the training section of the proposed algorithm. Considering the runtime, we cannot precisely compare the new algorithm's runtime with other algorithms because the runtime of the new algorithm is computed on different hardware than what is used in order to compute the runtime for other algorithms. However, the runtime of the new algorithm is approximately close to the best algorithms regarding the small hardware specification differences.

5. Conclusions and future works

In this paper a new method to generate better reference texts to be used in the multireference cosine text similarity algorithm is proposed. The multireference cosine algorithm shows good results in large collections of text documents. By using better reference texts, its accuracy will be improved significantly. The new approach shows reliable and promising results. The new algorithm can achieve close to the cosine text similarity algorithm's accuracy, but while consuming less computational power. Furthermore, the algorithm achieves better recall in comparison with other recent methods. The novel approach of the multireference cosine algorithm towards measuring the similarity between two documents using a reference text makes the achievement of such high recall feasible. As stated in Section 3.2, the reference texts are initiated and mutated using 3-grams with high Tf-idf scores. Consequently, the final reference text consists of several high Tf-idf score 3-grams that are highly beneficial in terms of detecting the similarity between two documents. This quality of this algorithm makes it more sensitive to the similarity between two documents and therefore it delivers high recall. Thus, as the new method has better recall than other state of the art methods, it is more suitable for practical application because the system could assure the detection of all DND documents.

In future works, this algorithm can be adapted to map-reduce programming frameworks to be used in big data distributed frameworks such as Hadoop. Finding new ways to combine multiple reference texts that are generated using different datasets in order to achieve better general accuracy for diverse applications is another interesting future work. Extending the method to be able to work with multiple languages could be considered as another good study to focus on.

References

[1] Broder AZ, Glassman SC, Manasse MS, Zweig G. Syntactic clustering of the web. Computer Networks and ISDN Systems. 1997; 29 (8-13): 1157-1166.

[2] Fetterly D, Manasse M, Najork M. On the evolution of clusters of near-duplicate web pages. Journal of Web Engineering 2003; 2 (4): 228-246.

[3] Henzinger M. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; Las Vegas, NV, USA; 2006. pp. 284-291.

[4] Manku GS, Jain A, Das Sarma A. Detecting near-duplicates for web crawling. In: Proceedings of the 16th International Conference on World Wide Web; Las Vegas, NV, USA; 2007. pp. 141-150.

[5] Xiao C, Wang W, Lin X, Yu JX, Wang G. Efficient similarity joins for near-duplicate detection. ACM Transactions on Database Systems (TODS) 2011; 36 (3): 15.

[6] Conrad JG, Guo XS, Schriber CP. Online duplicate document detection: signature reliability in a dynamic retrieval environment. In: Proceedings of the Twelfth International Conference on Information and Knowledge Management; Las Vegas, NVa, USA; 2003. pp. 443-452.

[7] Theobald M, Siddharth J, Paepcke A. Spotsigs: Robust and efficient near duplicate detection in large web collections. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; Las Vegas, NV, USA; 2008. pp. 563-570.

[8] Mohammadi H, Nikoukaran A. Multi-reference Cosine: A New Approach to Text Similarity Measurement in Large Collections. arXiv preprint arXiv:181003099. 2018.

[9] Ntoulas A, Cho J, Olston C. What's new on the web? The evolution of the web from a search engine perspective. In: Proceedings of the 13th International Conference on World Wide Web; Las Vegas, NV, USA; 2004. pp. 1-12.

[10] Croft WB, Metzler D, Strohman T. Search Engines: Information Retrieval in Practice. New York, NY, USA: Addison-Wesley Reading, 2010.

[11] Koppula HS, Leela KP, Agarwal A, Chitrapura KP, Garg S et al. Learning URL patterns for webpage de-duplication. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining; Las Vegas, NV, USA; 2010. pp. 381-390.

[12] Bar-Yossef Z, Keidar I, Schonfeld U. Do not crawl in the dust: different urls with similar text. ACM Transactions on the Web 2009; 3 (1): 3.

[13] Dasgupta A, Kumar R, Sasturkar A. De-duping URLs via rewrite rules. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; Las Vegas, NV, USA; 2008. pp. 186-194.

[14] Shivakumar N, Garcia-Molina H. SCAM: A copy detection mechanism for digital documents. In: Proceedings of the 2nd International Conference on Theory and Practice of Digital Libraries; Austin, TX; 1995. pp. 1-4.

[15] Manber U. Finding similar files in a large file system. Usenix Winter 1994; 94: 1-10.

[16] Heintze N, Labs B. Scalable document fingerprinting. In: 1996 USENIX Workshop on Electronic Commerce; Oakland, CA, USA; 1996. pp. 1-12.

[17] Brin S, Davis J, Garcia-Molina H. Copy detection mechanisms for digital documents. In: ACM Special Interest Group on Management of Data (SIGMOD) of the Association for Computing Machinery Record; Las Vegas, NV, USA; 1995. pp. 398-409.

[18] Indyk P, Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing; Las Vegas, NV, USA; 1998. pp. 604-613.

[19] Schleimer S, Wilkerson DS, Aiken A. Winnowing: local algorithms for document fingerprinting. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data; Las Vegas, NV, USA; 2003. pp. 76-85.

[20] Chowdhury A, Frieder O, Grossman D, McCabe MC. Collection statistics for fast duplicate document detection. ACM Transactions on Information Systems 2002; 20 (2): 171-191.

[21] Sarawagi S, Kirpal A. Efficient set joins on similarity predicates. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data; Las Vegas, NV, USA; 2004. pp. 743-754.

[22] Charikar MS. Similarity estimation techniques from rounding algorithms. In: Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing; Las Vegas, NV, USA; 2002. pp. 380-388.

[23] Pamulaparty L, Rao CG. A novel approach to perform document clustering using effectiveness and efficiency of simhash. International Journal of Engineering and Advanced Technology 2013; 2 (3): 312-315.

[24] Hajishirzi H, Yih Wt, Kolcz A. Adaptive near-duplicate detection via similarity learning. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval; Las Vegas, NV, USA; 2010. pp. 419-426.

[25] Lin J. Brute force and indexed approaches to pairwise document similarity comparisons with MapReduce. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval; Las Vegas, NV, USA; 2009. pp. 155-162.

[26] Vernica R, Carey MJ, Li C. Efficient parallel set-similarity joins using MapReduce. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data; Las Vegas, NV, USA; 2010. pp. 495-506.

[27] Pamulaparty L, Rao CG, Rao MS. A near-duplicate detection algorithm to facilitate document clustering. International Journal of Data Mining & Knowledge Management Process 2014; 4 (6): 39.

[28] Varol C, Hari S. Detecting near-duplicate text documents with a hybrid approach. Journal of Information Science 2015; 41 (4): 405-414.

[29] Zhang X, Yao Y, Ji Y, Fang B. Effective and fast near duplicate detection via signature-based compression metrics. Mathematical Problems in Engineering 2016; 2016: 1-12.

[30] Cilibrasi R, Vitányi PM. Clustering by compression. IEEE Transactions on Information Theory 2005; 51 (4): 1523-1545.

[31] Li M, Paul MP. An Introduction to Kolmogorov Complexity and Its Applications. New York, NY, USA: Springer Publishing Company, 2008.

[32] Liu H, Hu Z, Mian A, Tian H, Zhu X. A new user similarity model to improve the accuracy of collaborative filtering. Knowledge-Based Systems 2014; 56: 156-166.

[33] Ide N. The American National Corpus: Then, now, and tomorrow. In: Selected Proceedings of the 2008 HC-SNet Workshop on Designing an Australian National Corpus; Australia; 2009. pp. 108-113.

[34] Lang K. Newsweeder: Learning to filter netnews. In: Machine Learning Proceedings; Tahoe City, CA, USA; 1995. pp. 331-339.

[35] Cho J, Garcia-Molina H, Haveliwala T, Lam W, Paepcke A et al. Stanford WebBase components and appfcations. ACM Transactions on Internet Technology 2006; 6 (2): 153-186.