Developing a Technique for Overcoming the Searching Limitations of Documents

Chapter · January 2020

DOI: 10.1007/978-981-13-7564-4_59

M. Ali Akber Dewan
Athabasca University

Chapter 59
Developing a Technique for Overcoming the Searching Limitations of Documents

Md. Muntasir Shahriar, Mohammad Shamsul Arefin and M. Ali Akber Dewan

1 Introduction

Nowadays, we often need to search different documents for useful information. Searching the documents manually is very much time-consuming. At present, there are many searching techniques or software to search the documents based on users' query. The main target of the searching software is to find out information from the documents related to query information. The main limitation of this type of searching is that it cannot find results if there is no accurate match. Another limitation is that they fail to perform group search. If a match is found this type of searching returns "Found" and indicate the matching sequence. On the other hand, if it does not match accurately then it returns "Not Found". So, we need to write the accurate word or character sequence in order to find it out in the documents. For any kind of silly mistake like missing a semicolon or a space or missing a character form a paragraph, this type of tools returns that the information is not found although the information is in the document. There is also a big problem while searching for using synonyms. If we use synonym of any word in our search query, the system will return that the result is not found. These are the problems in current searching systems. Considering these facts, in this paper, we develop techniques that will overcome the limitations of the current searching techniques. We have developed the techniques for the documents of two different languages: English and Bangla. The paper is organized as follows.

Md. Muntasir Shahriar · M. S. Arefin (✉)
Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh
e-mail: sarefin@cuet.ac.bd

Md. Muntasir Shahriar
e-mail: m.shahriar44225@gmail.com

M. Ali Akber Dewan
School of Computing and Information Systems, Athabasca University, Edmonton, AB, Canada
e-mail: adewan@athabascau.ca

M. S. Uddin and J. C. Bansal (eds.), *Proceedings of International Joint Conference on Computational Intelligence*, Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-13-7564-4_59

In Sect. 2, we first discuss the previous work on offline document searching. We conclude this section by discussing the drawback of some existing offline document searching system. The overall system architecture, module and the algorithms are provided in Sect. 3. In this section, we also discuss about the steps of generating output. In Sect. 4, we presented the experimental results and evaluation of our system. Finally, we conclude the paper in Sect. 5.

2 Related Work

In recent years many works consider developing efficient searching techniques of documents. Previously no work has been done for overcoming the searching limitations of “Bangla” documents. In 2011, Arefin et al. [1] introduced statistical method for detecting plagiarism from Bangla and English electronic documents. Their proposed system can perform plagiarism checking in a Bangla documents from English documents and vice versa. It can also detect documents from same language. Singh et al. [2] developed the word searching algorithm (WSA) in a very efficient way in 2009. They are splitting the text into number of tables and match the hash value of the pattern with the Hash value of the words of the same length in the test. This algorithm is called modified word searching algorithm (MWSA). The experimental results of modified word searching algorithm (MWSA) was much faster than the previously proposed word searching algorithm (WSA) algorithm. Mishr and Pandey [3] used the word searching algorithm to search each appearance of the pattern by the brute force manner in each table. They proposed a word searching algorithm to reduce the number of comparisons to search the pattern in the offline text. Kujala and Elimma [4] optimize the cost of searching documents by using the Kolmogorov complexity of the request sequence. Thabit and Al-Ghuribi [5] proposed a searching algorithm where they used block and word prefix that is much faster than the binary search algorithm. Mishra et al. [6] improved the efficiency of binary search tree by adopting binary search tree and SDBM hash function techniques. Those techniques are applied to make first the search time to find a unique key for each pattern. In 2013, Ahmed et al. [7] analysis 880 status messages collected from a widely used social network sites and observed that unavailability and inadequacy information on web in developing countries play a significant role to motivate users using social network sites for information retrieval. They also try to emphasize the difference between social search and traditional web search. Mukhopadhyay et al. [8] proposed an improved snippet analysis based page ranking algorithm and support for image and news search. Chao [9] developed Dijkstra shortest path search algorithm through improving data structure. The results show that the developed dijkstra shortest path search algorithm can improve storage efficiency and reduce meaningless operation. Chen and Jia [10] proposed a new searching algorithm called diamond search. Simulation results demonstrate that their proposed algorithm performs better than another well-known fast block matching algorithm greatly.

3 System Architecture and Design

The system architecture of our system comprises three basic modules: document processing module, input processing module, and storage processing module. The function of document processing module is to extract all documents into a common format, so it will be easy for searching. Documents also divided into text query and numerical number in this module. Input processing module analyzes the input query and tokenizes it by using lexical analysis. Storage module stores all related information to the database and matching with the input query with documents. The architecture of our system is shown in Fig. 1.

3.1 Document Processing Module

The documents processing module consist of sub-modules: documents extraction, documents tokenization and store documents into database. In this module, we took several documents as input then tokenize them and store in the database. The functions of documents processing module are described as follows.

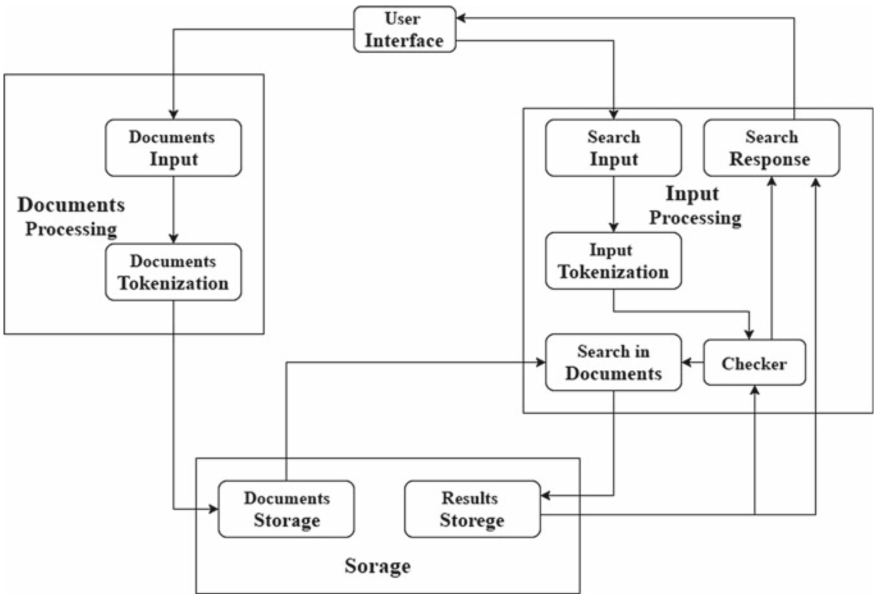

Fig. 1 System architecture of offline document searching system

3.1.1 Document Extraction

At first, all the documents are converted into a common type of document. So, it will be easy to perform searching operation. In our system, we extracted all the documents into PDF format. In this module, Bangla and English are separated. Numerical and text documents are also separated in this module. It will reduce searching time.

3.1.2 Document Tokenization

A document consists of sentences. A sentence is a set of words separated by a stop mark (".", "?", "।" or "!"). Tokenization is the process of breaking up the given text into units called tokens. The tokens may be words or number or punctuation mark. Tokenization does this task by locating word boundaries. Ending point of a word and beginning of the next word is called word boundaries. Tokenization is also known as word segmentation. If we consider a line "I am a student of CUET". After tokenization we get "I", "am", "a", "student", "of", "CUET".

3.1.3 Store Documents into Database

This module stores the documents from a user-specified location into database. The document database is created with document ID and document title. In storage, documents are stored. When we select documents from user interface then preprocessing will occur and tokens are stored into the database. Algorithm 1 shows the document tokenization process.

```
Algorithm 1: Documents tokenization and load into database
Input: Documents
Require: Store of documents
1.  Begin
2.  Documents are split into sentences
3.  For each sentence do
4.      Sentences are breaking down into tokens
5.      Store tokens into database
6.  End for
7.  End.
```

3.2 Input Processing Module

Input processing module consist of sub-module: search input, tokenization, checker, search result in documents and search response. We design this module for user of our system. A user can input his/her estimate query to search into documents and observe the result onto the screen. The functions of input processing module are given below.

3.2.1 Search Input

When we input a query for search into the document where we want to search, the query splitting into sentences and each sentence is breaking down into tokens. We develop the system for search in both single and multi-documents. So, we need to select that whether we want to search in single document of multi-documents at the same time. The tokens are then sends to the checker for further operation.

3.2.2 Checker

If we want to search same query which was searched before for the same document that will be waste of time. For reducing this unnecessary time, we develop a checker for our system. When tokens are forwarded to the checker with query id, it checks for the requested document in results storage. If it found match, then it returns result directly without searching in the document. That will save our valuable times. But if it does not find any match then forward the tokens for searching results in the documents. The algorithm of checker is shown on Algorithm 2.

Algorithm 2: Check the results for input query
Input: Input search query
Require: For checking whether the query was search before or not

```
1.  Begin
2.  For each selected item do
3.     Search into result_storage
4.  If query id and document id match
5.          Return info_id to the search result
6.     Else
7.          Forward input query for search in documents
8.  End if
9.  End for
10. End
```

3.2.3 Search Results in Documents

When the tokens are forwarded by the checker for searching in documents, each token are act like a separate unite. They search in documents individually and produce individual results. Then we need to select how our system generates result. If we want only the exact result, then we need to select Validate Exactness. The system will reassemble the tokens and match with the results which was found individually and return only exact result. We can also select the percentage of matching for generating results. For example, if we select 80% matching. Then our system display results which are above 80% match with the input query. Here we calculate the matching percentage with the length of the query.

$$\text{Percentage of Matching} = \left(\frac{\text{Length of Input Query}}{\text{Length of the indicated main document}}\right) * 100$$

For example, if we want to find out the sentence “I am a student of cuet” from the main document “I am a student and I read in cuet”, it will find out the sequential matching of words like “I am a student cuet” and calculate the percentage of matching.

3.2.4 Search Response

After getting results from the checker or results storage it’s term to display results on the screen. Generally, results are reached at the form of doc_id, page_no, and word_no. The responsibility for this function is to highlight the result of the targeted document with different color. As a result, it will be very helpful for finding results. If there exist many results on different pages, it will generate a link for each result. If we click on a link, it will return the estimated page on the screen where the result exists. So, we need not search on different pages manually for finding result. Algorithm 3 shows the procedure.

```
Algorithm 3: Search into documents
Input: Input search query
Required: Generating results for input
 1. Begin
 2. Initially set result empty
 3. For each selected token do
 4.     Search in documents
 5.  If found then
 6.         Update result
 7. Else
 8.          Go to step 3
 9. End if
10. End for
11. End
```

3.3 Storage Module

This module handles the storage of the documents and results for processing purpose. Documents storage store documents from user specified location to database. When anyone wants to search in a document which is already stored in the database, our system retrieves the document and execute the search operation. Documents are stored into the database as shown in Fig. 2.

When we obtain results for search query against any document it stores the results into the database. The result storage database is created with result ID, query and info ID, where info ID is connected with document ID and its position for that result. The result is then forwarded to the search response and checker. Search response highlighted the result into the document and display on the screen. And checker checks the results for new query whether the query was search before for resulted document or not. In Fig. 2 result ID and query of the result storage table represents the information about the result. Info ID and doc ID of the information table represents the position of the result in the document. And doc ID and content of the document storage table represents the documents.

After each searching, the result storage table is updated. Whenever a new search result is found a new raw is added to the result storage table. Every row consists of result_ID, query, doc_id, and info_ID. Algorithm 4 shows result updating technique.

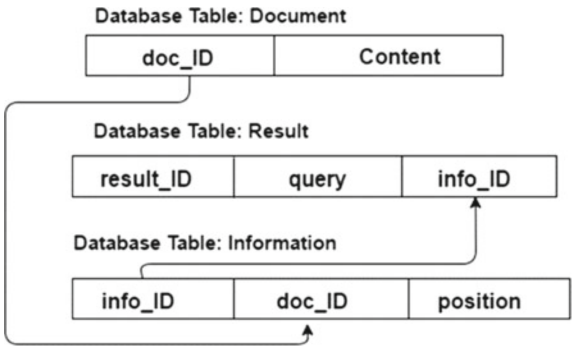

Fig. 2 Storage of document and result information

```
Algorithm 4: Updating result storage
Input: Searching results
Require: For storing result
1.  Begin
2.  For any search input do
3.      Search result in result storage
4.  If result is empty then
5.          Return output is empty
6.      End if
7.      For each raw in result do
8.           Update info_ID with raw.info_ID
9.            If new result_ID is found then
10.                  Add new raw in result storage
11.           Else
12.                  Do nothing
13. End if
14. End for
15. End for
16. End
```

4 Experimental Results and Evaluation

In this section, we provide a description of generating output and calculate the overall performance of our developed system. We measure our system performance by using precision and recall.

4.1 Output Generation

If we want to search in a document through our developed system, at first, we need to add a document where he/she want to search. Then we need to choose an option like exact search, case sensitive, case insensitive, multiward, spelling mistake or word missing search option. After that, we need to type query in search window and click on the search button. The output will be generated in a table and we can easily find out the result in the page by clicking on the result.

4.1.1 Exact Search Option

If we select the exact search option, our system will find out the exactly matching query from the document like other existing searching system. There is also an option for case sensitive and case insensitive for an exact search option. If we select case insensitive option, our system will not consider the upper-case and lower-case of the letter. But if we select case sensitive option, our system will find out the exact result. This is the common form of searching technique.

4.1.2 Multi-word Search Option

Multi-word search option is a very effective searching technique. By using this, we can search more than one item in a single search. So, it will save our valuable time. For this search option the input word or number are separated by space, comma, semicolon, exclamatory or dot sign. Then the separated items act like individual item and perform search operation individually. The main difference between exact search and multi-word search is, in exact search after searching individually all the tokens are combined to gather and find match according to their position. But in multi-word search, after searching individually it simply outputs the result without combining it.

4.1.3 Spelling Mistake Search Option

Spelling mistake search option is a very useful searching technique. If anyone makes mistakes in typing unwillingly or if he/she forgets the spelling of the search query our system will also able to find out the result. But there is a condition that the input query must not be more than two editing error. For this, our system will search for the related word in a word dictionary which is containing more than 27.9 million words. It will find out those words from the dictionary which have maximum of two editing distance. Among those words, it will start searching from the minimum editing distance word in our targeted document. If it finds any result, then stop searching for the next related word. So, although we make simple mistakes in spelling, our system will able to find out the results.

4.1.4 Missing Word Search Option

If we want to search for a sentence or a paragraph or a column in a document but unfortunately missing a few words from input query, then this option is very useful. Our system is able to find out the output for the missing words in input query. For this operation, our system split the input query into number of tokens. Then searching for the individual token in the document is performed. After that our system looks for the word sequence as like the input sequence. After finding the sequence, it calculates the matching percentage as below:

$$\text{Percentage of Matching} = \left(\frac{\text{Length of Input Query}}{\text{Length of the indicated main document}}\right) * 100$$

User can also select the percentage of matching for generating results. For example, if any user selects 80% matching then the system will generate results whose has above 80% match with the input query. Our system also performs search operation in more than one document at the same time. For this, we need to add document files as much as we want to execute for search operation. If we want to search for same item in several documents, our system able to search in all the documents at the same time.

4.2 Evaluating Performance

In this section, we have calculated the performance of our proposed system with respect to some tested documents. We have described the successful and unsuccessful cases of our system. We have also analyzed the performance of our developed system. The accuracy of our system is also calculated in this section. In our experiment, we used precision and recall for performance measurement of finding result from document as precision and recall are a better technique for finding effectiveness of data extraction system. By using precision and recall carve we evaluated the performance of our system with various options.

Precision is the fraction of relevant instances among the retrieved instances. It is also known as positive predictive value. It is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records extracted. The equation is:

$$\text{Precision} = \frac{\text{Relevent Documents}}{\text{Relevent Documents} + \text{Irelevent Documents}}$$

Recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. It is also known as sensitivity. It is the measurement of relevant records extracted by the system based on all relevant records that should be extracted by the system. The equation is

$$\text{Recall} = \frac{\text{Extracted Documents}}{\text{Extracted Documents} + \text{Not Extracted Documents}}$$

For calculating the performance of our system, we generated output for several inputs in several documents, and we also calculated the output for the same input manually. So, we can easily calculate the successful and unsuccessful case of our developed system. For big size documents, we use other software for finding results and compare with our system. From the successful and unsuccessful case, we can easily have calculated the precision and recall. And from precision and recall, we can also calculate the accuracy and quality of our entire system. If we multiply precision and recall with 100 we will get accuracy and quality. For measuring the performance, we generated results against 10 document files with our system and manually. Than

Table 1 Precision and Recall table for output results

File name	Size (kb)	Searching option	Item found	Item not found	Wrong found	Precision	Recall	Accuracy (%)	Quality (%)
FILE 1	82	Exact	5	0	0	1	1	100	100
		Multi	7	0	1	0.88	1	88	100
		Spelling mistake	5	0	1	0.83	1	83	100
		Word missing	1	0	0	1	1	100	100
FILE 2	137	Exact	6	0	0	1	1	100	100
		Multi	9	0	0	1	1	100	100
		Spelling mistake	6	0	0	1	1	100	100
		Word missing	1	0	0	1	1	100	100
FILE 4	471	Exact	21	0	0	1	1	100	100
		Multi	72	0	4	0.94	1	94	100
		Spelling mistake	24	0	9	0.72	1	72	100
		Word missing	1	0	1	0.50	1	50	100

(continued)

Table 1 (continued)

File name	Size (kb)	Searching option	Item found	Item not found	Wrong found	Precision	Recall	Accuracy (%)	Quality (%)
FILE 5	506	Exact	17	0	0	1	1	100	100
		Multi	50	0	4	0.93	1	93	100
		Spelling mistake	11	0	2	0.85	1	85	100
		Word missing	1	0	0	1	1	100	100
FILE 6	1123	Exact	8	0	0	1	1	100	100
		Multi	10	0	2	0.83	1	83	100
		Spelling mistake	5	0	2	0.71	1	71	100
		Word missing	2	0	0	1	1	100	100
FILE 7	1244	Exact	11	0	0	1	1	100	100
		Multi	98	0	25	0.80	1	80	100
		Spelling mistake	32	0	5	0.87	1	87	100
		Word missing	1	0	0	1	1	100	100

(continued)

Table 1 (continued)

File name	Size (kb)	Searching option	Item found	Item not found	Wrong found	Precision	Recall	Accuracy (%)	Quality (%)
FILE 8	1989	Exact	21	1	0	1	0.96	100	96
		Multi	59	0	2	0.97	1	97	100
		Spelling Mistake	70	0	8	0.90	1	90	100
		Word Missing	2	0	0	1	1	100	100
FILE 9	2141	Exact	15	0	0	1	1	100	100
		Multi	94	0	7	0.93	1	93	100
		Spelling mistake	60	0	21	0.74	1	74	100
		Word missing	5	1	0	1	0.83	100	100
FILE 10	5602	Exact	15	0	0	1	1	100	100
		Multi	94	0	7	0.93	1	0.93	100
		Spelling mistake	60	2	3	0.95	0.95	0.95	0.95
		Word missing	3	0	1	0.75	1	75	100

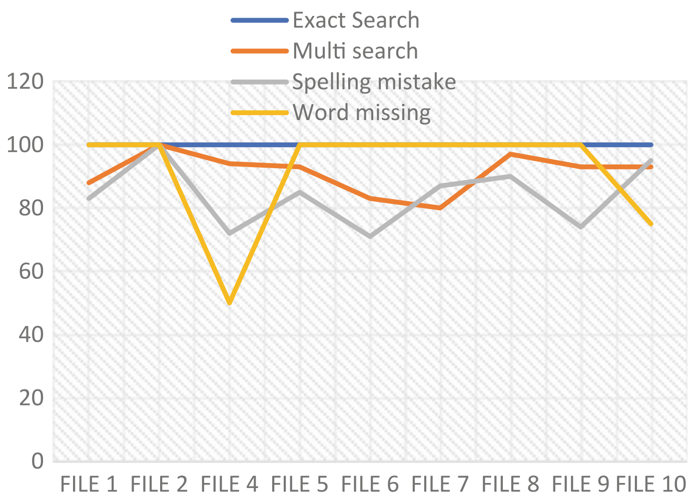

Fig. 3 Accuracy graph for output result

calculated precision, recall, accuracy, and quality of our system which is shown in Table 1.

The accuracy graph for exact search, multiword search, spelling mistake, and missing words is shown in Fig. 3.

Our system's performance is very well for exact search. We get 100% percent accuracy for exact search. But we got some incorrect results for multi-word search and spelling mistake search.

5 Conclusion

The searching limitations of documents searching are a great problem for big size documents like books. Considering this fact, in this paper, we proposed an approach to overcome the searching limitations of documents. Our system can perform search over multiple documents of two languages: Bangla and English. In addition, our system is able to perform group search. Although the current system can perform the search operation in the documents of two languages, we can modify it easily to apply it in multilingual contents.

References

1. Arefin MS, Morimoto Y, Sharif MA (2011) Bilingual plagiarism detector. In: Proceedings of the 14th international conference on computer and information technology, Dhaka, Bangladesh
2. Singh B, Yadav I, Agarwal S, Prassad R (2009) An efficient word searching algorithm through splitting hash the offline text. In: Proceedings of international conference on advances recent technologies in communication and computing, pp 6–9
3. Mishr SK, Pandey M (2010) An efficient word matching algorithm for offline text. In: Proceedings of international computer science and engineering conference, vol 10, pp 23–28
4. Kujala J, Elimma T (2007) The cost of offline binary search tree algorithms and the complexity of the request sequence. In: Proceedings of international conference on theoretical computer science, Tampere, Finland, pp 231–239
5. Thabit K, Al-Ghuribi SM (2013) A new search algorithm for documents using blocks and words prefix. Acad J 8(16):640–648
6. Mishra SK, Pandey M, Astya PN (2015) Efficient single pattern searching algorithm for offline test by using binary search tree. In: Proceedings of international conference on computing, communication and automation
7. Ahmed S, Anik TA, Tasnim M, Ferdous HS (2013) Statistical analysis and implications of SNS search in under-developed countries. In: Proceedings of international conference on advances on social network, pp 334–340
8. Mukhopadhyay D, Sharma M, Joshi G, Pagare T (2013) Experience of developing a meta-semantics search engine. In: Proceedings of international conference on cloud & ubiquitous computing & emerging technologies, pp 167–171
9. Chao Y (2010) A developed Dijkstra algorithm and simulation of urban path search. In: Proceedings of the 5th international conference on computer science and education, pp 1164–1167
10. Chen H, Jia H (2009) A newly developed diamond search algorithm. In Proceedings of the eighth IEEE international conference on dependable, autonomic and secure computing, pp 811–814
11. Knuth-Morris-Pratt algorithm. https://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm
12. Regular expression matching can be simple and fast. https://swtch.com/~rsc/regexp/regexp1.html
13. Approximate query matching. https://en.wikipedia.org/wiki/Approximate_query_matching
14. Lexical analysis. https://en.wikipedia.org/wiki/Lexical_analysis
15. Lai L, Wu C, Lin P, Huang L (2011) Developing a fuzzy search engine based on fuzzy ontology semantic search. In: IEEE international conference on fuzzy systems, pp 2684–2689
16. Sing B, Yadav I, Prasad R (2015) An efficient word searching algorithm splitting and hashing the offline text. In: Proceedings of international conference on advance in recent technologies in communication and computing, pp 2578–2582
17. Jain S, Panday M (2012) Hash table based word searching algorithm. In: Proceedings of international conference on advances on social network, pp 4385–438 (2012)