IJEEC

*International Journal of Electrical Engineering and Computing*
*Vol. 3, No. 1 (2019)*

***Original research paper***
UDC **519.857:004.021]:669.295.018.8**
DOI **10.7251/IJEEC1901001G**

# A three-phase mapreduce-based algorithm for searching biomedical document databases

**Milana Grbić**

Department of Mathematics and Computer Science, Faculty of Natural Science and Mathematics, University of Banja Luka, Republic of Srpska, Bosnia and Herzegovina

*milana.grbic@pmf.unibl.org*

*Abstract*—**Retrieving information from large document databases is in the focus of scientific research in recent years. In this paper, a parallel algorithm for searching biomedical documents based on the MapReduce technique is presented. The algorithm consists of three phases: preprocessing phase, document representation phase, and searching phase. In the first phase, lemmatization and elimination of stop words are performed. In the second phase, each of the documents is represented as a list of pairs (word, tf-idf index of the word). The third phase represents the main searching procedure. It uses a specially designed ranking criterion, which is based on a combination of the term frequency - inverse document frequency (tf-idf) index and the indicator function for each query word. Four different versions of ranking criteria are proposed and analyzed. The algorithm performances are tested on different subsets of the large and well-known PubMed biomedical document database. The results obtained by the experiments indicate that the proposed parallel algorithm succeeds in finding high-quality results in a reasonable time. Comparing to the sequential variant of the algorithm, the experiments show that the parallel algorithm is more efficient since it finds high-quality solutions in significantly less time.**

***Keywords- tf-idf index; mapreduce; parallel searching algorithm; biomedical documents;***

## I. INTRODUCTION

Over the past decades, a huge number of biomedical documents have been recorded. As a consequence, there is a growing need for the development of efficient software tools for searching related literature, such as scientific papers, review articles, and journal texts. Activities which include the development of such tools fall into a special area of information sciences called information retrieval. Information retrieval is finding information (e.g. texts or documents) from a large collection of data, that satisfies specific information queries [1] The information retrieval has a lot of applications, for example in managing digital libraries, developing search engines, media search and so on [2]. As a result of intensive research, a lot of new technologies and solutions have been developed in this field, such as web search engines, junk-email filters, news clipping services, etc. [3].

In this paper, the problem of finding relevant biomedical documents for a given set of query words is analyzed. At the beginning of the overall searching procedure, it is useful to find an appropriate form for document representation, which can make the searching process easier. Also, it is necessary to establish a criterion for determining the relevance of documents for the given words. Finally, it is required to check the documents from the corpus against the established criterion and perform the sort of documents by their relevance.

The main contribution of this paper is in adapting the MapReduce parallel programming technique for solving the problem of searching large biomedical document databases. After standard procedures of lemmatization and eliminating stop words, in the main searching phase, the proposed algorithm uses a specially designed function as the ranking criterion for determining the relevance of each document in the document database. After that, the list of documents, sorted by relevance is created.

The remainder of the paper is organized as follows. In the next section, the most relevant results related to the development of searching tools of biomedical documents are listed. In Section III the proposed algorithm is described in details. Section IV contains experimental results obtained on the subset of a PubMed biomedical database. The last section concludes the paper and proposes some directions for future work.

## II. LITERATURE REVIEW

Existing document searching tools can be classified into three categories:

i. those that perform the query only in the fields of citations;

ii. those that perform the query in the full-text article;

iii. those that further process the retrieved citations to organize them and/or to retrieve further information [1, 4].

One of the most popular services is the PubMed (available on https://www.ncbi.nlm.nih.gov/pubmed/), which is a part of

the National Center for Biotechnology Information (NCBI). PubMed is designed for searching abstracts by taking query words as an input, adding Boolean operators into the user queries and using automatic term mapping (ATM). Through its ATM process, PubMed service automatically compares and maps the words from the user query to the lists of pre-indexed terms (e.g. Medical Subject Headings MeSH). As a result of this search, PubMed retrieves documents containing the query terms and if the user query can be mapped into a MeSH concept, PubMed also retrieves documents indexed with those MeSH terms. The wide usage of the PubMed service caused that this service has been expanded for more specific purposes [5]. Some tools comparable to the PubMed, like RefMed, MedlineRanker, MiSearch, and iPubMed, base their search on keywords and/or look for particular citations, title, and authors. Searching documents with RefMed has a few iterations. First, RefMed bases its search of query words in the title and abstract of the document and returns a list of documents as the result. After that, it explicitly asks the user for feedback about the relevance of listed documents. RefMed uses the obtained answers for forming the ranking function. This procedure is subsequently repeated until the user receives satisfying results [6]. MedlineRanker is a web server which uses a given set of abstracts for learning the most discriminative words related to the topic. The obtained set of words is used for ranking new abstracts [7]. MiSearch forms the profile of the user by automatically saving information of citations viewed by the user during browsing. By using that profile, it calculates the rank of the future search results and places on the top those articles which are most likely to be seen by user [8]. Interactive PubMed (iPubMed) has two unique features: allows interactive and approximate search [9].

Another service, also maintained by NCBI, is the PubMed Central (https://www.ncbi.nlm.nih.gov/pmc/). Unlike the PubMed which is designed for searching abstracts, PubMed Central is designed for searching full texts. Since 2000. it has been available as a free archive of biomedical and life sciences journal literature at the U.S. National Institutes of Health's National Library of Medicine (NIH/NLM). Beside the PubMed Central, some other services for searching full biomedical texts are in common use, like eTBLAST and QUERTLE [1]. eTBLAST compares documents in the database with the input query and finds the documents that match best the keywords extracted from the query by analyzing the word alignment [10]. QUERTLE is a "relationship-driven biomedical search" tool, which performs queries based on the meaning and the context of documents [11].

Several information retrieval methods, like boolean queries and index structures, similarity queries and vector model and latent semantics indexing are presented in [12]. One of the crucial steps for finding information stored in literature is the term identification process. The overview of the state-of-the-art approaches dealing with this task is presented in [13].

There is a number of tools designed for searching specific queries. For example, the information extraction system for locating protein-protein interaction data and collecting those data in Biomolecular interaction network database (BIND) has been presented in [14]. Hoffman and Valencia [15] proposed the so-called iHOP (Information Hyperlinked over Proteins) web service that uses genes and proteins as hyperlinks between the sentence and abstracts, which enables better navigation through information from PubMed. The comparison between metadata and full-text searching for gene names in two biomedical literature domains is presented in [16].

To our knowledge, current solutions do not completely address the specific demand of prioritizing some query words in the searching process. The aim of our work is to develop such a searching algorithm, which enables this specific searching task.

## III. THE THREE-PHASE PARALLEL SEARCHING ALGORITHM

### *A. Problem definition*

Searching biomedical document databases belongs to a class of information retrieval (IR) problems, particularly to the problems of searching document libraries. The enormous growth of biomedical data and the need for finding relevant data with specific purposes, like biocuration, in vitro experiments and gene annotations, make this problem specific to other IR problems. It is well known that biomedical data are Big data and they can be broken by 3V characteristics: Volume, Velocity, and Variety. Therefore, big data technologies are increasingly used for biomedical informatics research and the problem of searching biomedical documents can be a matter of specific investigation. Although there is a number of sophisticated searching tools available, to our knowledge, a searching task which involves the relevance of specific words has not been adequately addressed. The problem considered in this paper deals with this challenge and the solution can improve the efficiency of specific searching requests which arises in biomedical science.

Let a document database be given. The input for the problem is the list of query words. Optionally, the level of importance of each query word can be assigned by a numerical value. The greater numerical value means the more importance given to the word. The task is to find the most relevant documents that contain given query words, where the searching process is guided by a specific ranking function. The output of the algorithm is the list of documents containing the query words and sorted by the ranking criteria.

In a more practical explanation, to the user is offered a simple interface which allows the input of query words which are the subject of the search. The user can further define the level of importance of each query word by assigning a numerical value to each word. As it is mentioned, the greater numerical value means the more importance of a word. After the user set the input data, the algorithm enters into the searching phase, which will be explained in details in the following sections. As a result, the user gets a list of documents containing the input words. The list is sorted by the criteria which will be discussed at the end of this section.

### *B. Parallel computing environment*

In recent years there are a fast growth parallel computing environments, like multi-core, many-core, GPU or cluster frameworks. Two most commonly used cluster computing frameworks are Apache Spark on Hadoop and Open MP/MPI [17]. Apache Spark is a platform developed at UC Berkley that exploits in-memory computation for solving iterative algorithms. The advantage of this platform is that it can be run in traditional clusters such as Hadoop [18]. OpenMP/MPI efficiently exploits multi-core clusters architectures such as

Beowulf. It combines the MPI (Message Passing Interface) paradigm with shared memory multiprocessing. As it is stated in [17], Spark efficiently deals with fault tolerance support and data replication, but it has a clear impact on the speed. On the other side, OpenMP/MPI provides a solution mostly oriented to high-performance computing but vulnerable to faults. Spark on Hadoop framework offers a distributed file system with failure and data replication management and allows the addition of new nodes at runtime. In our paper, we use the advantage of the Spark platform, since it provides a set of tools for data analysis and management that is easy to use and deploy.

As it is already mentioned, Spark is based on the concept of maintaining data in memory rather than on disk. It is able to efficiently deal with iterative computational procedures that recursively perform operations over the same data. Resilient Distributed Datasets (RDDs) are fundamental data units in the Spark environment. Formally, RDD is a distributed memory abstraction that provides in-memory computation on large clusters in a fault-tolerant manner. There are two ways of creating RDD. The first one is to apply some deterministic operations on data in stable storage while the second one is to apply these operations on other RDDs. By default, the Spark keeps RDD in the memory, but if there is not enough RAM it can split them to the disk [19].

The MapReduce programming model is one of the most successful implementations for processing and generating large data sets. In the core of the MapReduce model, there is a map function that processes key/value pairs to generate a set of the intermediate key/values pairs. These intermediate key/values pairs are the input for the reduce function, which merges all intermediate values associated with the same intermediate key [20]. This programming paradigm has already been proven as a successful technique for processing big datasets of clinical, biomedical, and biometric data [21]. To our knowledge, this paradigm has not been used for the information retrieval problem, especially in the field of biomedical documents.

The proposed searching algorithm consists of three phases. In the first phase, which is described in more details in Subsection C, data preprocessing is performed. In the second phase (Subsection D), documents are transformed into a more suitable and informative form. And finally, in the third phase, described in Subsection E, documents search based on the set of query words is done.

### *C. The data preprocessing phase*

Data preprocessing is a common starting step in the text mining process. It includes several standard procedures, such as lemmatization and eliminating stop words [22]. In the beginning, all characters from the set
{'.', '(', ')', '/', '%', '-', ',', ';', ':', '*', '[', ']', '#', '+', '\ ', '\$', '?', '!', ""'}
are identified so they can be removed in the process of eliminating stop words. After that, all letters are converted into the lower case. All words are the further subject of lemmatization process, i.e. the process of identifying basic forms of each word [23]. For the lemmatization of the text, WordNetLemmatizer (nltk.stem package available at http://www.nltk.org/api/nltk.stem.html) is used in this paper. Before eliminating standard stop words in English, the list of stop words is updated with punctation and some other special signs. In the proposed MapReduce algorithm, all these actions are performed using the mapValues function.

### *D. The representation phase*

The task of the second phase is to represent each document as a list of pairs (word, tf-idf index of the word). The term frequency - inverse document frequency (tf-idf) represents a statistical measure of the importance of a word for a given document from a corpus of documents. tf-idf is a commonly used metric with a property that the higher value of tf-idf index means the stronger connection between the word and the document [24]. For a given word t and a document d in a corpus of documents *D*, *tf-idf* is calculated as

$$tf\text{-}idf\ (t, d, D) = tf\ (t, d) * idf\ (t, D) \quad (1)$$

where *tf(t, d)* is the number of occurrences of the word *t* in document *d*, while the value *idf* may be calculated by the formula

$$idf\ (t, D) = log\ \frac{|D|+1}{df(t,D)+1} \quad (2)$$

The value *df (t,D)* is the number of different documents in the corpus *D* which contain the word *t*. |*D*| is the total number of documents in the corpus *D*.

### *E. The searching phase*

After representing each document in the form of the corresponding list of pairs, the algorithm enters the main phase - searching for the most relevant documents. Let $t_1,t_2,\ldots, t_n$ be query words and let *d* be a document from the corpus *D*. The relevance of document *d* for the words $t_1,t_2,\ldots, t_n$ is calculated by the formula

$$f(t_1, t_2, \ldots, t_n, d, D) = \sum_{i=1}^{n}(w_i * tf\text{-}idf(t_i, d, D)) + \alpha * \sum_{i=1}^{n} I(t_i, d) \quad (3)$$

As can be seen from the Eq. (3), the overall ranking criterion is a combination of two metric functions. The first one is the sum of *tf-idf* indices for each query word. In addition for each query word $t_i$ the value *tf-idf* $(t_i,d,D)$ is multiplied by the weighted factor $w_i$. This approach enables the ranking criterion to be more robust since each query word can be given different importance in the overall search. If each weight is equal to 1, then each word is given the same importance. The second part of the ranking criterion is the sum of indicator function $I(t_i,d)$ which takes value 1 if the word $t_i$ is present in document *d*, 0 otherwise. It should be noted that the sum of indicator functions is multiplied by binary parameter $\alpha$, taking values from {0,1}. Actually, if $\alpha = 1$ indicator functions are taken into the account, otherwise not.

Finally, if the value of the function *f* from the Eq. (3) for the document $d_1$ is greater than the value of that function for the document $d_2$, then the document $d_1$ is more relevant for the given query words than document $d_2$.

Actually, by the Eq. (3), four different types of ranking criteria can be given:

- Ranking criterion I: $w_i$ =1, for all *i=1,...,n*, and $\alpha = 1$ : all query words have the same importance and indicator functions are taken into the account;
- Ranking criterion II: $w_i$ =1, for all *i=1,...,n*, and $\alpha = 0$ : all query words have the same importance and indicator functions are omitted;

- Ranking criterion III: the weights of query words are different and $\alpha = 1$ : query words are given different importance and indicator functions are taken into account;
- Ranking criterion IV: the weights of query words are different and $\alpha = 0$ : query words are given different importance and indicator functions are omitted.

Parameter variation allows the user to influence the searching procedure. If the parameter $\alpha$ is equal to 1, then the more favored space of the search is the subset of documents which contains all the query words. On the other side, if the parameter $\alpha$ is set to 0, then the requirement that the documents contain all query words is not so strict. By using weights of the query words, the user can additionally influence the search results. If all the query words should have the same importance, then all the weights should be the same (by default they are set to 1). Otherwise, if some query words are more important than others, the user can assign greater weights to them. As a consequence, the existence of those words in the documents will more influence on the overall value of the ranking function.

Although weights of the query words in Eq.(3) can be arbitrary assigned, they should be chosen in such a way that the algorithm still stays stable. Preliminary experiments indicated that higher precision of the search is achieved if the value of function *f* for ranking criteria III and IV is not significantly different from the function *f* for ranking criteria I and II. In our experiment, the weights are chosen from the range [0,1] and values of function *f* are slightly less than in the case where all weights are equal to 1.

## IV. EXPERIMENTAL RESULTS

In order to examine the characteristics of the proposed algorithm, comprehensive experiments have been performed. For all tests, Intel i7-4770 CPU@3.40GHz with 8GB RAM and Windows 7 operating system was used. For implementing MapReduce functions in the Spark-Hadoop environment, Python programming language was used. All searching queries were executed on subsets of the PubMed document database which is publicly available and can be downloaded from the address ftp://ftp.ncbi.nlm.nih.gov/pub/pmc.

The proposed algorithm was tested on four subsets of the PubMed database of different sizes: PubMed200, PubMed1000, PubMed5000, and PubMed10000, containing approximately 200, 1000, 5000 and 10000 documents, respectively. Each of these subsets was created by expanding the previous smaller subset. An exact number of documents and the total size of each subset are shown in Table 1. All documents are in the txt format. In all experiments the following set of 12 query words was used:

{'nucleic', 'acid', 'polarization', 'atomic', 'electrostatic', 'biological', 'experimental', 'rna', 'backbone', 'force', 'center', 'md'}.

TABLE 1. Total number of documents

| Subset | Number of doc. | Size (in MB) |
|---|---|---|
| PubMed200 | 185 | 7.74 |
| PubMed1000 | 1086 | 51.1 |
| PubMed5000 | 4801 | 166 |
| PubMed10000 | 10389 | 354 |

### *A. Experiment with ranking criterion I*

In the main experiment, each query word is given the same importance (for all *i=1,...,n* the values $w_i$ are set to 1), indicator functions are included ($\alpha = 1$) and each of four document subsets is used. For the execution of the proposed parallel algorithm, six worker nodes in Spark-Hadoop environment are used.

#### *1) Results obtained on the PubMed200 subset*

Table 2 contains the list of the 10 most relevant document when the smallest subset PubMed200 was used as the test database.

The third column contains the value *f* obtained for a document and given query words. For example, the value 189.52 in the first row (document #1) is calculated by summing *tf-idf* indices from the list of the pairs (word, *tf-idf* index)

[('center', 1.16), ('rna', 50.32), ('md', 38.13),('force', 25.55), ('electrostatic', 7.71),('experimental', 10.88),('acid', 4.80), ('atomic', 7.25), ('polarization', 6.23), ('backbone', 15.65), ('biological', 2.11), ('nucleic', 7.71)],

TABLE 2. Results on the PubMed200 subset

| | Document | Value *f* |
|---|---|---|
| 1 | Acc_Chem_Res_2010_Jan_19_43(1)_40-47 | 189.52 |
| 2 | Acc_Chem_Res_2014_Jun_17_47(6)_1731-1741 | 162.76 |
| 3 | Acc_Chem_Res_2014_Sep_16_47(9)_2837-2845 | 82.01 |
| 4 | PMC5406124 | 77.95 |
| 5 | Acc_Chem_Res_2012_Jul_17_45(7)_1122-1131 | 76.99 |
| 6 | Acc_Chem_Res_2014_Sep_16_47(9)_2812-2820 | 70.09 |
| 7 | Acc_Chem_Res_2014_Oct_21_47(10)_3118-3126 | 64.92 |
| 8 | Acc_Chem_Res_2012_Aug_21_45(8)_1258-1267 | 63.04 |
| 9 | Acc_Chem_Res_2012_Dec_18_45(12)_2035-2044 | 60.81 |
| 10 | Acc_Chem_Res_2014_Jun_17_47(6)_1825-1835 | 58.86 |

and the number 12, since all query words are present in the document Acc_Chem_Res_2010_Jan_19_43(1)_40-47. On the other side, document #10 (Acc_Chem_Res_2014_-Jun_17_47(6)_1825-1835) contains only 6 query words with the following *tf-idf* indices

[('acid', 10.86), ('backbone', 0.63), ('rna', 3.00),

('biological', 1.20),('nucleic', 36.47), ('electrostatic', 0.70)]

most of which indices are smaller than for the previously considered document #1.

*2) Results obtained on the PubMed1000 subset*

The list of the top 10 documents obtained on the subset PubMed1000 is shown in Table 3.

The highest value of the function *f* in the subset PubMed1000 is obtained for the document PMC5333189 and this value is significantly higher than for other top documents. The main reason is that the query word 'rna' appears more than 550 times in that document, so tf-idf index of the word 'rna' is very high. Comparing the results from Table 3 to the results obtained on the subset PubMed200, it can be seen that the most relevant documents found in the PubMed200 are now at positions 3 and 6.

*3) Results obtained on the PubMed5000 subset*

In this section, the experiments are extended to the subset which contains about 5000 documents. Top ten most relevant documents from this document subset (PubMed5000) are shown in Table 4. By comparing these results with the results obtained on the PubMed1000 subset, one can see that the first 5 documents are the same. Also, it can be seen that documents which are in the smaller database at positions 6 and 7 are now at positions 8 and 9.

As an illustration of the behavior of the function *f*, we compare *tf-idf* indices of the same document in different subsets. At Fig.1 we show *tf-idf* indices of the document PMC5371978, which appears at the position 8 in the subset PubMed1000 and at the position 6 in the subset PubMed5000. From Fig.1 it can be seen that for each query word, except the word 'md', *tf-idf* index is higher in the case of the subset PubMed5000 than of the subset PubMed1000. Different values of these indices are a consequence of the fact that the proportion of the *df* measure of a word, in the overall collection *D* is, in general, lower if the collection *D* is larger, and thus the overall *idf* and *tf-idf* measures are higher (Eq. (1), (2)).

TABLE 3. Results on the PubMed1000 subset

| | **Document** | **Value *f*** |
|---|---|---|
| 1 | PMC5333189 | 462.71 |
| 2 | ACS_Nano_2011_May_24_5(5)_3405-3418 | 239.96 |
| 3 | Acc_Chem_Res_2010_Jan_19_43(1)_40-47 | 182.73 |
| 4 | ACS_Chem_Biol_2013_Dec_20_8(12)_2697-2706 | 175.25 |
| 5 | ACS_Nano_2014_May_27_8(5)_4771-4781 | 165.02 |
| 6 | Acc_Chem_Res_2014_Jun_17_47(6)_1731-1741 | 147.67 |
| 7 | ACS_Nano_2014_Aug_26_8(8)_7620-7629 | 143.04 |
| 8 | PMC5371978 | 130.71 |
| 9 | ACS_Nano_2014_May_27_8(5)_4559-4570 | 127.84 |
| 10 | ACS_Nano_2015_Oct_27_9(10)_9731-9740 | 126.46 |

TABLE 4. Results on the PubMed5000 subset

| | **Document** | **Value *f*** |
|---|---|---|
| 1 | PMC5333189 | 595.63 |
| 2 | ACS_Nano_2011_May_24_5(5)_3405-3418 | 309.29 |
| 3 | Acc_Chem_Res_2010_Jan_19_43(1)_40-47 | 235.15 |
| 4 | ACS_Chem_Biol_2013_Dec_20_8(12)_2697-2706 | 218.81 |
| 5 | ACS_Nano_2014_May_27_8(5)_4771-4781 | 207.88 |
| 6 | PMC5371978 | 198.92 |
| 7 | Acta_Crystallogr_D_Biol_Crystallogr_2013_Nov_1_69(Pt_11)_2174-2185 | 198.19 |
| 8 | Acc_Chem_Res_2014_Jun_17_47(6)_1731-1741 | 187.97 |
| 9 | ACS_Nano_2014_Aug_26_8(8)_7620-7629 | 182.09 |
| 10 | ACS_Nano_2011_Feb_22_5(2)_693-729 | 181.04 |

*4) Results obtained on the PubMed10000 subset*

In Table 5 we show top 10 results obtained on the PubMed10000 subset. As in two previous cases, the document PMC5333189 is again the most relevant. The second and the third results are also the same as in the case of the PubMed5000 subset. From Tables 4 and 5, one can see that several other documents appear in both tables but in different orders.

*B. Justification of the proposed ranking criterion I*

In order to further examine the performances of different ranking criteria proposed in Subsection III-E, some additional experiments are performed. The algorithm is executed on the subset PubMed1000 three more times, once for each of the proposed ranking criteria II, III and IV. All the ranking criteria are calculated by varying parameters in Eq. (3). For $\alpha = 1, w_i = 1$ for all $i$ the first ranking criterion is formed. The second one (all query words have the same importance and indicator functions are omitted) is obtained for $\alpha = 0$ and $w_i = 1$ for $i=1,...,n$. In the third and the fourth criteria the weights are different and are defined as follows:

[('nucleic',0.3), ('acid', 0.5), ('polarization', 0.2),('atomic', 0.2), ('electrostatic',0.1),('biological',0.4),( 'experimental', 0.1), ('rna', 0.7), ('backbone', 0.6), ('force', 0.3), ('center',0.1), ('md',0.01)].

In the third criterion $\alpha = 1$, while in the fourth $\alpha = 0$. Table 6 contains some comparative results obtained by each of these four ranking criteria: the value of the function $f$ (shown in the column Value $f$) and the position in the ranking list (column Pos.). From Table 6 it can be seen that the first and the second most relevant documents are the same for all four considered criteria. The document, which is on the third place in cases of the first and the second criteria, is at positions 11 and 12 for the other two criteria. That is a consequence of the fact that the words with larger tf-idf indices are given relatively small weights. This is how we can influence the overall search in cases when we want to give more importance to some specific query words. In addition, by varying the value $\alpha$ we can further influence the search results by favoring those documents which contain all or most of the query words.

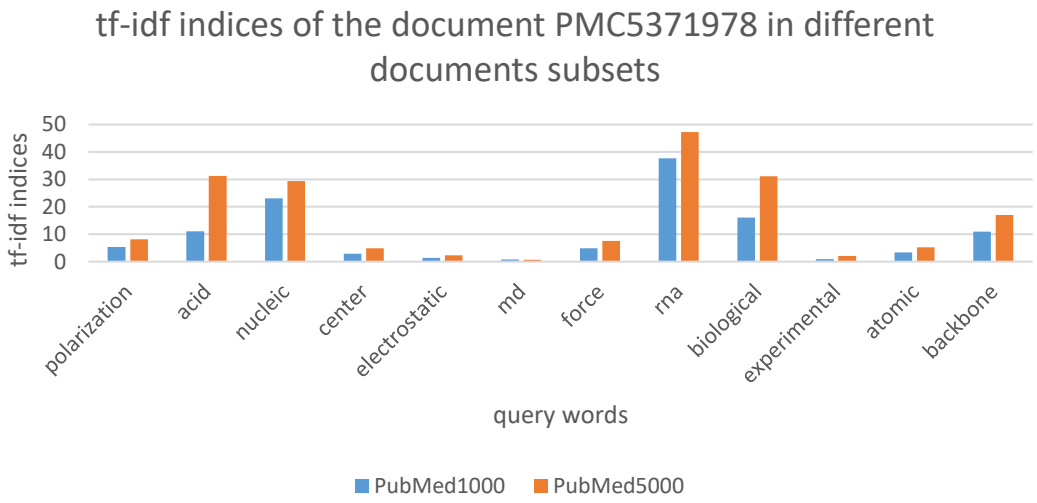


Figure1. tf-idf indices of the document PMC5371978 in different documents subsets

TABLE 5. Results on the PubMed10000 subset

| | Document | Value f |
|---|---|---|
| 1 | PMC5333189 | 557.43 |
| 2 | ACS_Nano_2011_May_24_5(5)_3405-3418 | 289.16 |
| 3 | Acc_Chem_Res_2010_Jan_19_43(1)_40-47 | 241.5 |
| 4 | Acta_Crystallogr_D_Biol_Crystallogr_2013_Nov_1_69(Pt_11)_2174-2185 | 212.22 |
| 5 | PMC5371978 | 211.05 |
| 6 | ACS_Nano_2016_Jul_26_10(7)_7117-7124 | 206.45 |
| 7 | ACS_Nano_2011_Feb_22_5(2)_693-729 | 205.07 |
| 8 | ACS_Chem_Biol_2013_Dec_20_8(12)_2697-2706 | 204.57 |
| 9 | ACS_Nano_2014_May_27_8(5)_4771-4781 | 191.81 |
| 10 | Acc_Chem_Res_2014_Jun_17_47(6)_1731-1741 | 174.92 |

### *C. Justification of using a parallel algorithm*

In order to justify the usage of MapReduce paradigm the sequential algorithm is also developed and the performances of the parallel and sequential variants are compared. Table 7 contains execution times of both parallel and sequential algorithms for different document subsets. In the first column, the name of each subset is shown. In the last two columns, total execution times of parallel and sequential algorithms are shown, respectively, with the mark N/A in the case when sequential algorithm could not find a solution in a reasonable time. It should be noticed that obtained ranking lists of both algorithms are the same. From Table 7 one can easily see that the parallel algorithm is significantly faster than the sequential one: for the smallest set about 20 times, for the subset PubMed1000 about 80 times, while for the set containing about 5000 documents the parallel algorithm is faster more than 150 times.

## V. CONCLUSION AND FUTURE WORK

Developing practical methods for searching biomedical documents is of great interest in the scientific community. In this paper, a three-phase parallel searching algorithm is proposed. The algorithm is implemented in the Spark-Hadoop platform using the MapReduce paradigm. The searching criterion is specially designed as a combination of the *tf-idf* index with weighted factors and indicator functions. The benefit of this approach is that the user can influence the overall searching process by setting the parameters of the ranking function. The algorithm allows the user to prefer some query words by assigning larger weights to them. In addition, by setting the binary parameter $\alpha$ to 1 or 0, the user can influence the search results in the sense whether the most relevant documents should contain all query words or not.

The proposed algorithm is tested on the well-known PubMed biomedical document database. Experimental results clearly indicate the high usability of the proposed algorithm. Firstly, the algorithm succeeds to find satisfactory results in a reasonable time. Secondly, the obtained results are consistent with regards to the size of the document database. And finally, we show how to influence the final results by choosing different values of the ranking function parameters, as it is demonstrated in Subsection IV-B.

This research can be extended in several ways. In order to decrease needed memory space, it could be useful to represent documents in a more efficient way by using some hashing technique and adopt ranking criteria to deal with such representation. Our parallel algorithm could also be combined with cluster-based approaches to browsing large document collections. The purpose of this hybridized approach could be the speeding up the searching process and the reducing of the overall searching space by considering only particular cluster representatives.

TABLE 7 The time of execution

| **Subset** | **Time (in minutes)** | |
|---|---|---|
| | **Parallel** | **Sequential** |
| PubMed200 | 0.72 | 14.6 |
| PubMed500 | 3.1 | 266.14 |
| PubMed5000 | 14 | 2180.32 |
| PubMed10000 | 45 | N/A |

## ACKNOWLEDGMENT

This research was partially supported by Ministry for Scientific and Technological Development, Higher Education and Information Society, Government of Republic of Srpska, B&H, under the project "Development and application of combinatorial optimization and machine learning methods in bioinformatics" (2019).

TABLE 6. Results on the PubMed1000 obtained by using different ranking criteria

| **Document** | **Crit. I** | | **Crit. II** | | **Crit. III** | | **Crit. IV** | |
|---|---|---|---|---|---|---|---|---|
| | **Value *f*** | **Pos.** | **Value *f*** | **Pos.** | **Value *f*** | **Pos.** | **Value *f*** | **Pos.** |
| PMC5333189 | 462.71 | 1 | 453.71 | 1 | 306.06 | 1 | 297.06 | 1 |
| ACS_Nano_2011_May_24_5(5)_3405-3418 | 239.96 | 2 | 230.96 | 2 | 158.22 | 2 | 149.22 | 2 |
| Acc_Chem_Res_2010_Jan_19_43(1)_40-47 | 182.73 | 3 | 170.73 | 3 | 69.01 | 11 | 57.01 | 12 |
| ACS_Chem_Biol_2013_Dec_20_8(12)_2697-2706 | 175.25 | 4 | 164.25 | 4 | 103.01 | 4 | 92.01 | 4 |
| ACS_Nano_2014_May_27_8(5)_4771-4781 | 165.02 | 5 | 157.02 | 5 | 112.69 | 3 | 104.69 | 3 |
| Acc_Chem_Res_2014_Jun_17_47(6)_1731-1741 | 147.7 | 6 | 138.69 | 6 | 95.88 | 6 | 86.88 | 6 |
| ACS_Nano_2014_Aug_26_8(8)_7620-7629 | 143.04 | 7 | 136.03 | 7 | 96.25 | 5 | 89.25 | 5 |
| PMC5371978 | 130.71 | 8 | 118.71 | 9 | 67.65 | 13 | 55.65 | 13 |
| ACS_Nano_2014_May_27_8(5)_4559-4570 | 127.84 | 9 | 120.84 | 8 | 89.52 | 7 | 82.52 | 7 |
| ACS_Nano_2015_Oct_27_9(10)_9731-9740 | 126.46 | 10 | 118.46 | 10 | 84.96 | 8 | 76.96 | 8 |
| ACS_Nano_2011_Feb_22_5(2)_693-729 | 121.03 | 11 | 111.03 | 13 | 42.54 | 19 | 32.54 | 21 |
| ACS_Nano_2015_Jan_27_9(1)_251-259 | 119.71 | 12 | 113.71 | 11 | 81.79 | 9 | 75.79 | 9 |
| ACS_Chem_Biol_2015_Mar_20_10(3)_652-666 | 105.34 | 15 | 96.34 | 15 | 68.05 | 12 | 59.05 | 11 |

## REFERENCES


[1] A. Manconi, E. Vargiu, G. Armano, & L. Milanesi. "Literature retrieval and mining in bioinformatics: state of the art and challenges." Advances in bioinformatics, 2012, 2012. URL: https://www.hindawi.com/journals/abi/2012/573846/

[2] R. Baeza-Yates & B. Ribeiro-Neto. "Modern information retrieval", volume 463. ACM press New York, 1999.

[3] A. Singhal. "Modern information retrieval: A brief overview." IEEE Data Eng.Bull., 24(4):35-43, 2001. URL: http://sifaka.cs.uiuc.edu/course/410s12/mir.pdf

[4] A. K Bajpai, S. Davuluri, H. Haridas, & G. Kasliwal, "In search of the right literature search engine (s)" Technical report. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.853.6711&rep=rep1&type=pdf

[5] Z. Lu. "PubMed and beyond: a survey of web tools for searching biomedical literature." Database, 2011, 2011. URL: https://academic.oup.com/database/article/doi/10.1093/database/baq036/460587

[6] H. Yu, T. Kim, J. Oh, I. Ko, S. Kim, & W. S. Han "Enabling multi-level relevance feedback on PubMed by integrating rank learning into DBMS" In BMC bioinformatics (Vol. 11, No. 2, p. S6). BioMed Central, , (2010, April). URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-S2-S6

[7] J. F. Fontaine, A. Barbosa-Silva, M. Schaefer, M. R. Huska, E. M. Muro & M. A. Andrade-Navarro "MedlineRanker: flexible ranking of biomedical literature" Nucleic acids research, 37(suppl_2), W141-W146, (2009) URL: https://academic.oup.com/nar/article/37/suppl_2/W141/1136707.

[8] J. David, A. S. Ade, Z. C. Wright, V. B. Aaron, & B. D Athey "MiSearch Adaptive PubMed Search Tool" Bioinformatics, (2008). URL: https://academic.oup.com/bioinformatics/article/25/7/974/209803

[9] J. Wang, I. Cetindil, S. Ji, C. Li, X Xie, G. Li & J Feng. "Interactive and fuzzy search: a dynamic way to explore MEDLINE" Bioinformatics, (2010), 26(18), 2321-2327 URL: https://academic.oup.com/bioinformatics/article/26/18/2321/208288

[10] J. Lewis, S. Ossowski, J. Hicks, M. Errami, & H. R Garner."Text similarity: an alternative way to search MEDLINE." Bioinformatics, 22(18):2298-2304, 2006. URL: https://academic.oup.com/bioinformatics/article/22/18/2298/318080

[11] P. Coppernoll-Blach. "Quertle: the conceptual relationships alternative search engine for pubmed". Journal of the Medical Library Association: JMLA, 99(2):176, 2011. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3066589/

[12] H. Shatkay & R. Feldman. "Mining the biomedical literature in the genomic era: an overview." Journal of computational biology, 10(6):821-855, 2003. URL:
http://research.cs.queensu.ca/home/shatkay/papers/JCB03.pdf

[13] M. Krauthammer & G. Nenadic. "Term identication in the biomedical literature" Journal of biomedical informatics, 37(6):512-526, 2004. URL:
https://www.sciencedirect.com/science/article/pii/S1532046404000826

[14] I. Donaldson, J. Martin, B. De Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G. D. Bader, K. Michalickova, et al. "PreBIND and Textomy -mining the biomedical literature for protein-protein interactions using a support vector machine." BMC bioinformatics, 4(1):11, 2003. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-4-11

[15] R. Hoffmann and A. Valencia. "Implementing the iHOP concept for navigation of biomedical literature." Bioinformatics, 21(suppl_2):ii252-ii258, 2005. URL: https://www.ncbi.nlm.nih.gov/pubmed/16204114

[16] B. M Hemminger, B. Saelim, P. F Sullivan, and T. J Vision." Comparison of full-text searching to metadata searching for genes in two biomedical literature cohorts." Journal of the Association for Information Science and Technology, 58(14):2341{2352, 2007. URL: https://ils.unc.edu/bmh/pubs/Comparison_of_Full-Text_Searching_to_Metadata_Searching-JASIST-2007.pdf

[17] Jorge L Reyes-Ortiz, L. Oneto, and D. Anguita. "Big data analytics in the cloud: Spark on hadoop vs mpi/openmp on beowulf. "Procedia Computer Science, 53:121-130, 2015. URL: https://ac.els-cdn.com/S1877050915017895/1-s2.0-S1877050915017895-main.pdf?_tid=2647e086-b7cf-40ab-86ec-943fbf9ed7e4&acdnat=1551874338_612707b3940ac1285a2478d8de07bb0e

[18] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker & I. Stoica "Spark: Cluster computing with working sets", HotCloud, (2010).,10(10-10), 95. URL: http://static.usenix.org/events/hotcloud10/tech/full_papers/Zaharia.pdf

[19] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J Franklin, S. Shenker, and I. Stoica. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pages 2-2. USENIX Association, 2012. URL: https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

[20] J. Dean and S. Ghemawat. "MapReduce: simplied data processing on large clusters." Communications of the ACM, 51(1):107{113, 2008.
URL:https://www.usenix.org/legacy/events/osdi04/tech/full_papers/dean/dean.pdf

[21] E. A. Mohammed, B. H. Far, & C. Naugler, "Applications of the MapReduce programming framework to clinical big data analysis: current landscape and future trends", BioData mining, (2014), 7(1), 22. URL: https://biodatamining.biomedcentral.com/articles/10.1186/1756-0381-7-22

[22] S Vijayarani, J Ilamathi, and Ms Nithya. "Preprocessing techniques for text mining-an overview". International Journal of Computer Science & Communication Networks, 5(1):7-16, 2015. URL: https://www.ijcscn.com/Documents/Volumes/vol5issue1/ijcscn2015050102.pdf

[23] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola. "Stemming and lemmatization in the clustering of finnish text documents" In Proceedings of the thirteenth ACM international conference on Information and knowledge management, pages 625-633. ACM, 2004. URL: https://tampub.uta.fi/bitstream/handle/10024/66142/stemming_and_lemmatization_in_the_clustering_2004.pdf?sequence=2&isAllowed=y

[24] J. Ramos. "Using tf-idf to determine word relevance in document queries." In Proceedings of the first instructional conference on machine learning, volume 242, pages 133-142, 2003. URL: https://www.semanticscholar.org/paper/Using-TF-IDF-to-Determine-Word-Relevance-in-Queries-Ramos/b3bf6373ff41a115197cb5b30e57830c16130c2c


**Milana Grbić** received her MSc degree (2016) in Mathematics at the Faculty of Mathematics at the University of Belgrade. She is a Ph.D. student in the field of data mining in bioinformatics. Her research interests include data classification, data mining, and bioinformatics.