Knowledge Engineering and Data Science (KEDS)
Vol 4, No 2, December 2021, pp. 105–116

pISSN 2597-4602
eISSN 2597-4637

Similarity Identification of Large-scale Biomedical Documents using Cosine Similarity and Parallel Computing

Merlinda Wibowo [a, 1, *], Christoph Quix [b, 2], Nur Syahela Hussien [c, 3], Herman Yuliansyah [d, 4], Faisal Dharma Adhinata [a, 5]

[a] *Faculty of Informatics, Institut Teknologi Telkom Purwokerto, Jl. DI Panjaitan No.128, Karangreja, Purwokerto, Indonesia*

[b] *Information Systems & Data Science, Hochschule Niederrhein, Adlerstraße 35, 47798 Krefeld, Germany*

[c] *Universiti Kuala Lumpur Malaysian Institute of Information Technology (UniKL MIIT) 1016, Jln Sultan Ismail, Bandar Wawasan, 50250 Kuala Lumpur, Malaysia*

[d] *Informatics Department, Universitas Ahmad Dahlan Jl. Kapas No.9, Semaki,Umbulharjo, Yogyakarta, Indonesia*

[1] *merlinda@iitelkom-pwt.ac.id*;* [2] *christoph.quix@hs-niederrhein.de;* [3] *syahela@unikl.edu.my;* [4] *herman.yuliansyah@tif.uad.ac.id;* [5] *faisal@ittelkom-pwt.ac.id*

** corresponding author*

ARTICLE INFO

Article history:
Submitted 7 December 2021
Revised 25 December 2021
Accepted 29 December 2021
Published online 31 December 2021

Keywords:
Biomedical Documents
Cosine Similarity
Keyword Extraction
Large Scale
Parallel Computing
Similarity Identification

ABSTRACT

Document similarity computation is an important research topic in information retrieval, and it is a crucial issue for automatic document categorization. The similarity value is between 0 and 1, then the closest value to 1 is represented both documents is considered more relevant, vice versa. However, the large scale of textual information has created the problem of finding the relevance level between documents. Therefore, the relevance between mesh heading text in the PubMed documents is higher than the relevance of the abstract text in the PubMed documents. Furthermore, parallel computing is implemented to speed up the large-scale documents similarity identification process that automatically calculates in the PubMed application. The execution time of mesh heading is 15.447 seconds, and the timely execution of abstract is 74.191 seconds. The execution time of mesh heading is higher than abstract because abstract contains more words than mesh heading. This study has successfully identified the similarity between large-scale biomedical documents of the PubMed documents that implemented a cosine similarity algorithm. The result has shown that the cosine similarity of the mesh heading texts is higher than the abstract text in the form of a graph and table shown in the PubMed application. The cosine similarity is useful to measure the similarity between documents based on the TF*IDF calculation result.

I. Introduction

The number of articles added to the literature databases is proliferating. Large amounts of textual data could be collected as a part of the research, such as scientific literature, transcripts in the marketing and economic sectors, speeches in political discourse, such as presidential campaigns and inauguration speeches, and meeting transcripts [1]. PubMed dataset of MEDLINE also has grown enormously [2]. This large amount of textual information has created the problem of finding the relevance level between documents. Besides, it has become challenging to manage and exploit them. This difficulty is closely related to the semantic aspect of these documents. A large amount of data brings about new opportunities for discovering new values, helps to gain an in-depth understanding of hidden values, and incurs new challenges such as how effectively organized and recognized data character [3][4]. There are two main parts for identifying PubMed documents to overcome the challenges. The two parts are abstract and Medical Subject Heading (Mesh) heading. Mesh heading

https://doi.org/10.17977/um018v4i22021p105-116

is the thesaurus for indexing, cataloging, and searching biomedical and health-related information. The relevance between mesh heading text in the PubMed documents is higher than the relevance of the abstract text in the PubMed documents. Besides, the National Library of Medicine provides the mesh heading.

Text mining in big data analytics is emerging as a powerful tool for harnessing the power of unstructured textual data by analyzing it to extract new knowledge and to identify significant patterns and correlations hidden in the data [1][5]. Furthermore, quickly detecting similar documents becomes a fundamental problem as times go on [6]. This difficulty is closely related to the semantic aspect of these documents. Indeed, manual operation is possible and gives good results. However, a manual procedure is not possible with a large corpus. Therefore, document similarity computation is an important research topic in information retrieval, and it is a crucial issue for automatic document categorization. Moreover, parallel computing (for big data) reduces the processing time and quickly detects similar documents [7][8]. Thus, the parallelization of big data is emerging as an essential framework for large-scale parallel data applications.

Some research determines the similarity between text used extracted keywords generated based on term frequency-inverse document frequency (TF*IDF) [9][10][11][12]. This research focuses on detecting the similarity of the document. The method for calculating similarity is cosine similarity then the result demonstrates that cosine similarity can calculate the difference of text document. Keyword extraction is a vital algorithm to extract appropriate keywords that can easily choose which document to read to learn the relationship between documents in the form of document retrieval, web page retrieval, document clustering, summarization, text mining, and others. It will automatically identify terms that best describe the keywords of a document [2][9][13]. Then, to obtain a suitable text relevance algorithm to demonstrate relevance calculation between two documents, many studies have been implemented the cosine similarity [9][14][15]. The cosine similarity is useful to measure the similarity between documents based on the result of the keyword extraction. However, the large-scale documents are needed extra time execution. Therefore, parallel computing is implemented to enhance the computing speeds by running several different tasks simultaneously on the same data [7][8]. Parallel computing refers to the breaking process of a more significant problem into smaller, independent parts. Often it can be executed concurrently by multiple processors communicating via shared memory then the results are combined upon completion as part of the overall algorithm. The main purpose of parallel computing is to increase the available computing power for faster application processing and troubleshooting.

This research aims to develop a text mining application that adapts a text similarity algorithm for the biomedical domain to identify the relationship and relevance between large-scale documents. The implemented algorithms are run on a set of the published article from the biomedical documents to which keyword annotations by experts exist to compare with automatically extracted keywords by a parallel computing engine.

II. Methods

In this study, the similarity identification framework provided a guideline to conduct and organize the research properly. The framework illustrated in Figure 1 showed the workflow divided into several research phases that describe the action plan step by step as a guide to complete this study. Each phase will require the output to ensure that the research goals are achieved successfully.

A. Master Data

PubMed is an open-access search engine launched in January 1996 and made freely available online one year and a half years later. It has become one of the most commonly used search tools for retrieving scientific data. An almost continuous increase in the performed searches has been observed in Biomedical and Life Sciences [2][16][17][18]. PubMed is a search tool provided by the United States National Library of Medicine (NLM). MEDLINE is a central bibliographic database maintained by the United States National Library of Medicine (NLM), is the most commonly used electronic database in applied, systematic reviews of biomedical research. It covers articles published from 1946 to the present, primarily in a scholarly journal. This database is freely accessible via the PubMed website for 24 million records. The sample of PubMed documents is depicted in Figure 2.

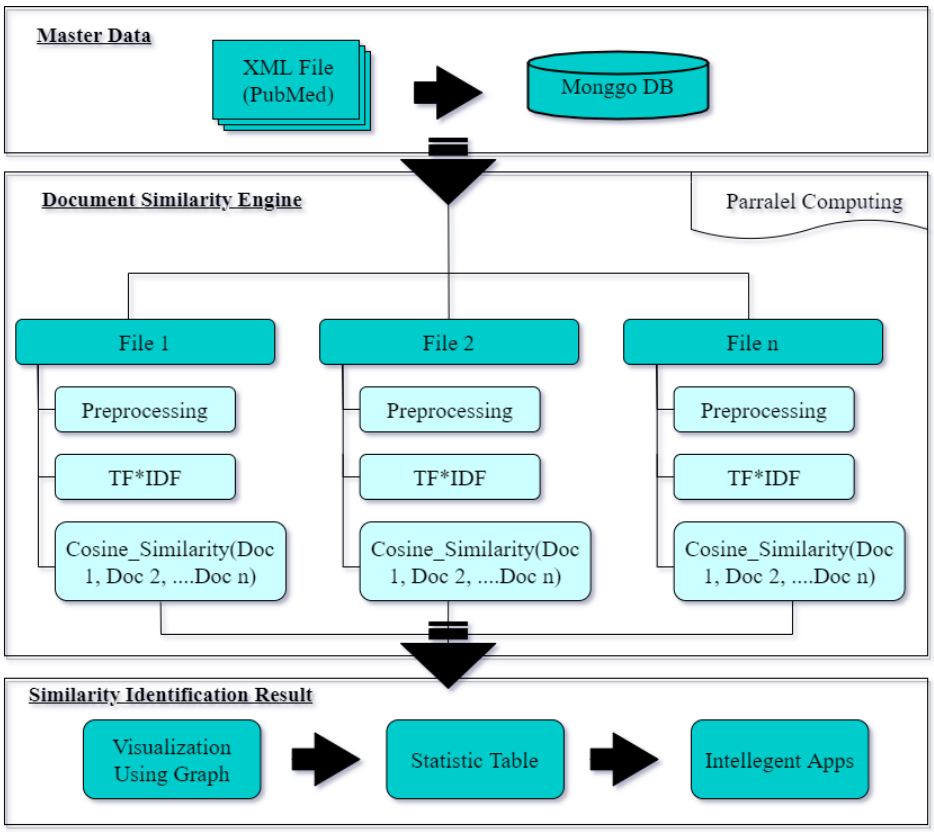
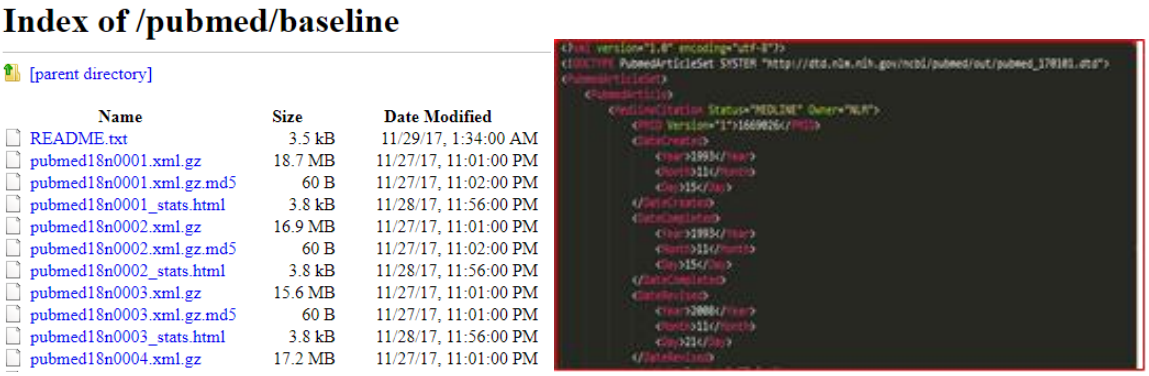


Fig. 1. The Similarity identification framework



(a) (b)

Fig. 2. (a) Sample image of PubMed document, and (b) The image of a data set represented in XML format

Figure 2(a) depicts the sample image of PubMed Document, and Figure 2(b) shows the dataset represented in the XML format. Each XML file consists of different publication articles; more than three thousand articles are in every XML file. Dataset will be stored in MongoDB to support the parallel computing process for document similarity identification. MongoDB is the most popular NoSQL database system [19].

MongoDB is a cross-platform document-oriented database system. As a NoSQL database, MongoDB avoids traditional table-based relational database structures that support JSON documents with dynamic schemes, making data integration in some application types easier and faster. Data is stored in a document consisting of key and value with type and size variable (not set before). Figure 3 illustrates the sample of the PubMed documents stored in MongoDB. The data successfully inserted in MongoDB will be used for the following process. This dataset will be in JSON format inside the MongoDB collection with the same tag as data in XML format. This tag can be used for reading the data for the following process. MongoDB does not use the query to read the data like a SQL database.

Fig. 3. Sample of PubMed documents stored in MongoDB

B. Documents Similarity Engine

Machine learning is a type of artificial intelligence that can learn from the data without explicit instructions and follow the instructions programmed [4]. Machine learning will assist in finding a solution optimizing performance by using sample data or previous experience to gain new insights, reveal new patterns, and produce more accurate results. This research will implement machine learning in the documents similarity engine to identify the similarity between large-scale documents known as master data by automatically extracting keywords using node.js. JavaScript is a programming language that runs on the client or browser side only, then Node.js exists to complete the JavaScript role. It can also apply as a programming language running on the server-side, like PHP, Ruby, or Perl. With parallel computing, the process will reduce the processing time and quickly detect the relationship and relevance between large-scale documents.

1) Preprocessing

At this stage, the results obtained from the master data will automatically go through to preprocess. The tag used in this study is Mesh Heading and Abstract. Both of the tags can represent the entire contents of the article published as testing data. This preprocessing will reduce the number of words that exist by removing stopwords and changing the words into the basic form (stemming) [9][20]. Stopword is words that are not a feature or unique word of a document like conjunctions. Taking into stopword in-text transformation will make the whole text mining system depend on the language factor. Therefore, it is a weakness of the stopword removal process. However, the stopword removal process is still used because this process will significantly reduce the system workload. By removing the stopword of a text, the system will only consider the considered important words.

Stemming reduces derived words to their word stem, base, or basic form. One of the most widely used stemming algorithms is the Porter Stemmer [9][20]. The process of treating words with the same stem as synonyms, e.g., query expansion for search engines, is called conflation. The stem does need not be identical to the morphological root of a word since, for purposes of conflation, it is usually sufficient that related words map to the same stem even if this stem is not in itself a valid root. For example, the preprocessing depicts in Figure 4.

*2) Representative Algorithm: TF*IDF*

This phase is representative of algorithm TF*IDF. The TF*IDF-statistic short for term frequency times inverse the document frequency can extract keywords from a document by considering a single document and all documents from the corpus [2][21]. The promising candidate for a keyword in a specific document if it shows up relatively often within the document and rarely in the rest of the

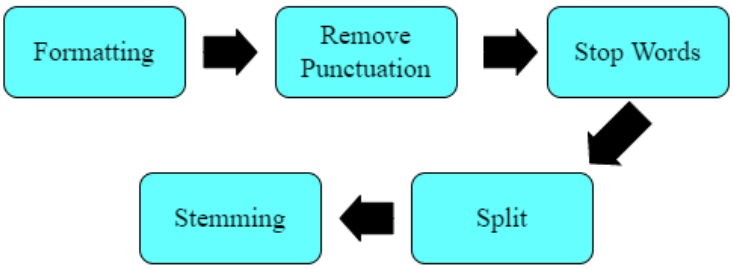


Fig. 4. Preprocessing

corpus is a word in the term of TF*IDF. The term frequency is given by the ratio of the number of term occurrences in the document and the number of occurrences of the most frequent word in one document. The formula of TF*IDF is shown in equation (1).

$$TF * IDF = \frac{freq(P,D)}{size(D)} . log_2 \left(\frac{N}{df(P)}\right) \quad (1)$$

where *freq(P,D)* is the number of times *P* occurs in document *D*, *size(D)* is the number of words in document *D*, *df(P)* is the number of documents containing *P* in the global corpus, and *N* is the size of the global corpus.

3) Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them [9][14][15]. Cosine Similarity measures the similarity between two vectors in a dimensional space obtained from the cosine value of the angle from the product of the two vectors being compared because the cosine of 0° is 1 and less than 1 for other angles values. The similarity value of the two vectors is similar when the value of cosine similarity is 1.

Cosine similarity is used in positive space, where the result is limited between values 0 and 1. If the value is 0, then the document is similar. If the result is 1, then the value is said to be dissimilar [9][14][15]. This limit applies to some dimensions. Therefore, cosine similarity is most often used in high-dimensional positive spaces. For example, in Information Retrieval, each term is assumed to be a different dimension. Furthermore, the document is marked with a vector where each dimension corresponds and how many terms appear. Equation (2) depicts the formula of cosine similarity.

$$\text{similarity} = \cos(\theta) = \frac{A \,.B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i \, B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2)$$

where A_i and B_i are components of vectors *A* and *B*. *A* is the weight of each feature in vector A. B is the weight of each feature in B. If it is associated with information retrieval, then A is the weight of each term in document A, and B is the weight of each term in document B. In this study, cosine similarity is used because large-scale PubMed documents are high-dimensional data. In large-scale PubMed documents that contain many published articles, it also can be said that each document consists of many different tags. Measurement of similarity can be done by comparing document 1 with document 2 then the system will calculate the similarity value. $A_i.B_i$ is the value obtained from term *A* and term *B*, then the two values are added together. The value of A_i^2 is all values of term-document *A*, and all values are raised to the power of two, and term B_i^2, all values obtained are raised to the power of two, then all values obtained are added up.

C. Similarity Identification Result

In this stage, the identification results of document similarities will be represented in a graph, statistical table, and web application. The visualization data using a graph and statistical table are intended to make it easier to present and understand the result [4][22]. Meanwhile, web application development can enhance the end-user experience and real-time data collection and provide custom content [22]. This study will show the graph and statistical table in the web application after the document similarity engine process has finished. For example, the PubMed Application interface web application depicts in Figure 5. The documents will be uploaded to the application. The application will automatically calculate the similarity between biomedical documents with parallel computing,

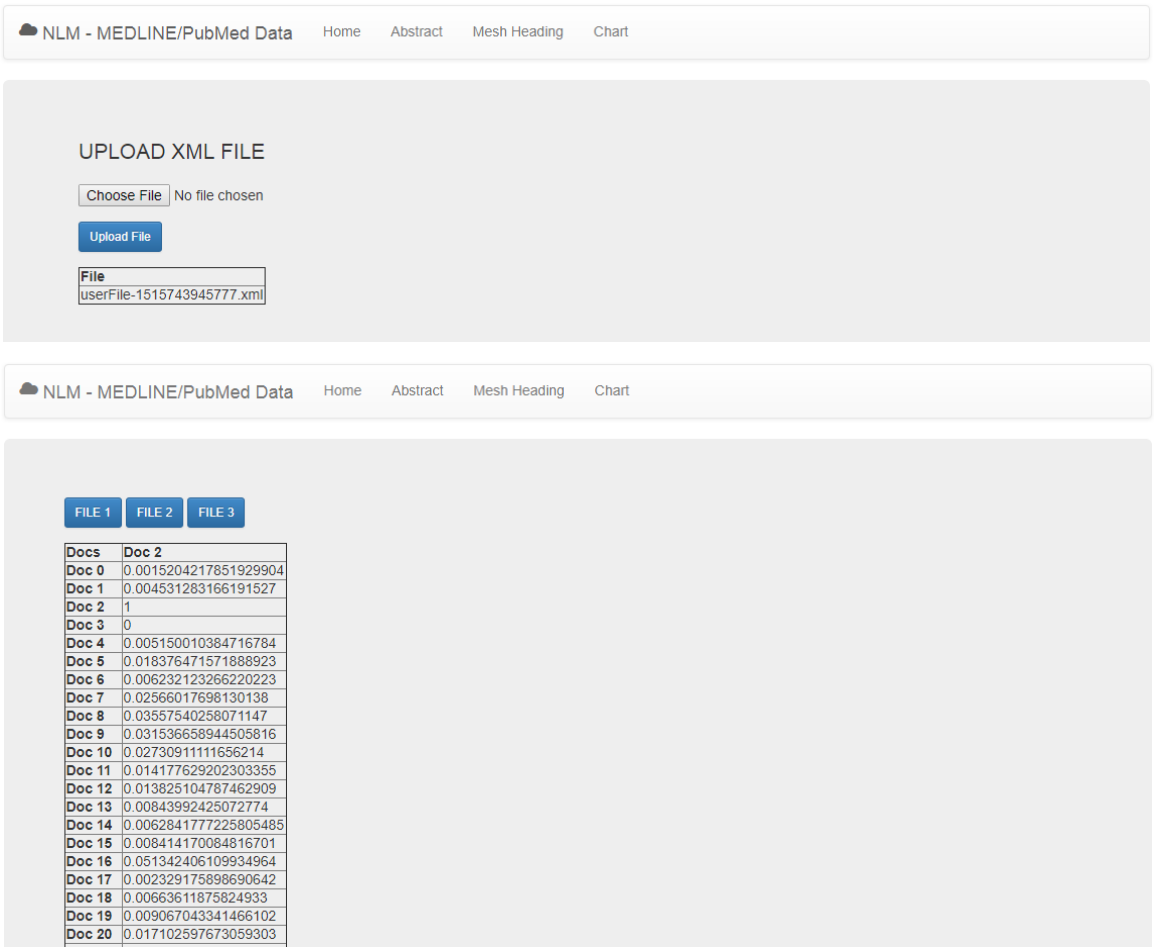


Fig. 5. PubMed application

reducing the processing time and quickly detecting the relationship and relevance between large-scale documents. Therefore, the results will be in the form of a graph and table that facilitate reading the calculation results.

III. Results and Discussions

The PubMed application developed as an identification documents similarity engine as an intelligent application that automatically calculated the similarity between biomedical documents then visualized the identification result in the form of a graph and table. The calculation process is used parallel computing that is reduced the processing time and quickly detects the relationship and relevance between large-scale documents. The first process is storing the master data in MongoDB. Then the punctuation will be removed, converted to lower case, implemented stop word removal, and extracted the basic word using the Porter Stemming algorithm. Two tags were used in this study, abstract and mesh heading. This tag can be used to read the data for the next process. Figure 6 depicts the sample abstract dataset from PubMed publications captured from MongoDB. In addition, the captured dataset is then transformed into the basic word. The basic word is the biomedical word, including the chemical formulation, medicine name, and others. Therefore, this need is needed to be considered.

Fig. 6. Sample captured abstract dataset

The listing program to get the extracted keywords can be seen in preprocessing program. The input in preprocessing program is all abstract data, and the output is the string of each word from the abstract. The first step of preprocessing is removing all conjunction and punctuation in the abstract then transforming the letter into lowercase. The next step is stemming the words into their roots.

Preprocessing program

Input: *abs_all*
Output: *all_string*
Initialization var *abs_all, all_string, removed_conjuction, text_array, reg, rm_punctutation, reg*

```
    removed_conjuction ← abstrak_fix.replace(regex_rm_conjuction," ")
    text_array ← removed_conjuction.replace(/(\s)?\d\s+/g, ' ').replace(/\n+/g,'
         ').split(" ").filter((d) => { return d != '' &&
         conjuction_list.indexOf(d.toLowerCase()) < 1
    }).map((d) => {
    reg ← new RegExp(/\d/,'gi')
    rm_punctuaction ← d.replace(regex_rm_punctuaction,'')
    return reg.test(d) ? d : stemmer.stem(rm_punctuaction)
         })
```

The sample of extracted keywords result is depicted in Figure 7.

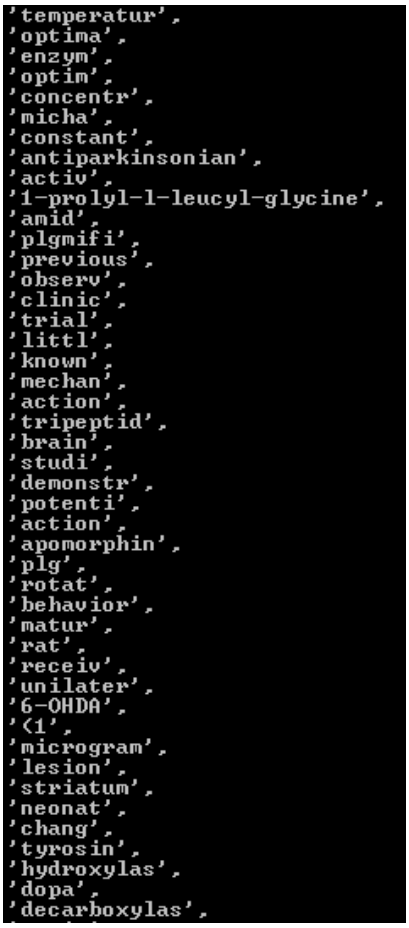


Fig. 7. Sample of extracted keyword results

Afterward, the extracted keyword weighting is carried out to calculate the frequency of occurrence of each word of the testing document in each document in the dataset. This phase is representative of algorithm TF*IDF. The TF*IDF can extract keywords from a document by considering a single document and all documents from the corpus. Finally, the TF*IDF calculation result is used to calculate the similarity of the documents testing with the PubMed documents using the cosine similarity algorithm. The listing program to get the term frequency value can be seen in the TFIDF program.

```
TFIDF program
Input: all_string
Output: tf
Initialization var all_string, tfidf, tf

   TfIdf ← natural.TfIdf
   tfidf ← new TfIdf()
        abs_all.forEach((dataa) => {
        tfidf.addDocument(dataa)
        })
   all_string.forEach((as) => {
        tfidf.tfidfs(as, function(i, measure) {
        })
```

The sample of TF*IDF results stored in MongoDB is captured in Figure 8.

Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in 0 and 1. This similarity calculation will result in a value between 0 and 1. The closer value to 1, then both documents are more related, vice versa.

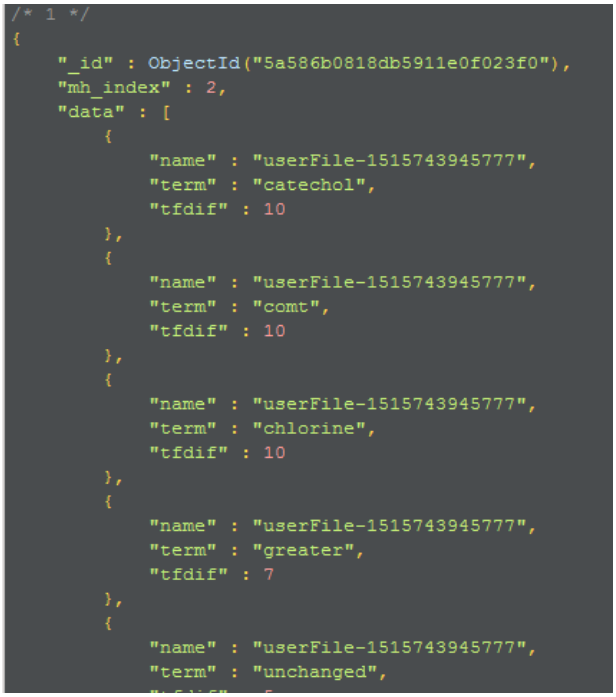


Fig. 8. Sample of captured TF*IDF results

From the similarity process that has been done, the cosine similarity produces similarity values between one document compared to other documents. The document comparison focused on the Abstract and Mesh Heading tag of the PubMed publications document as the testing data. The listing code to measure the cosine similarity between documents can be seen in the cosine similarity program.

Cosine_similarity program

Input: *tf*
Output: *cos_sim*
Initialization var *tf, cos_sim_all, l1, l2, tf1, tf2, sum, a, b, A, B, cos_sim, len_avg, len_avg2, tf_sum*

```
    l1 ← tf[item.first].length
    l2 ← tf[item.second].length
    tf1 ← tf[item.first]
    tf2 ← tf[item.second]
    if ( l1 > l2 ) {
    len_avg ← l1-l2
    for (var j=0; j<len_avg; j++){
    tf2.push({term : '-', tfdif : 0}) }}
    else{
        len_avg2 ← l2-l1
        for (var k=0; k<len_avg2; k++){
        tf1.push({term : '-', tfdif : 0}) }}
        tf_sum ← []
    tf1.forEach((item) => {
    a ← tf2.filter((d) => {
    return item.term == d.term && item.term != '-' && d.term != '-'})
    if (a.length > 0) {
    b ← item.tfdif*a[0].tfdif
    tf_sum.push(b) }})
    sum ← tf_sum.length > 0 ? tf_sum.reduce((accumulator, currentValue) => accumulator +
        currentValue) : 0
    A ← tf1.map((data, index) => {return Math.pow(data.tfdif,2)}).reduce((accumulator,
        currentValue) => accumulator + currentValue)
    B ← tf2.map((data, index) => {return Math.pow(data.tfdif,2)}).reduce((accumulator,
        currentValue) => accumulator + currentValue)
    Cos_sim← sum / (Math.sqrt(A)*Math.sqrt(B))
```

The cosine similarity results shown in Figure 9 illustrated the sample result of cosine similarity between abstract text with different abstracts in other publications and mesh heading text with the different mesh heading in other publications. For example, the cosine similarity between document 2 and document 1 between the mesh heading of published articles in the PubMed documents is 0.0045 and indicates that the cosine similarity is 0.45%.

Figure 10 illustrates the result of cosine similarity measurement between documents. In this case, it is using abstract and mesh heading text in each PubMed document. The graph of the cosine similarity result from this PubMed document is shown the mesh heading texts cosine similarity is higher than the abstract text. The results showed that the relevance between mesh heading text in the PubMed documents is higher than the relevance of the abstract text in the PubMed documents. Hence, the relationship and correlation between published articles in PubMed documents can be known from the mesh heading text. The number of words and terms in the abstract can affect text similarity results. Besides, this mesh heading tag can be used for subsequent data processing, such as classifying or clustering the PubMed documents.

Mesh Heading

Docs	Doc 2
Doc 0	0.0015204217851929904
Doc 1	0.004531283166191527
Doc 2	1
Doc 3	0
Doc 4	0.005150010384716784
Doc 5	0.018376471571888923
Doc 6	0.006232123266220223
Doc 7	0.02566017698130138
Doc 8	0.03557540258071147
Doc 9	0.031536658944505816
Doc 10	0.02730911111656214
Doc 11	0.014177629202303355
Doc 12	0.013825104787462909
Doc 13	0.00843992425072774
Doc 14	0.0062841777225805485
Doc 15	0.008414170084816701
Doc 16	0.051342406109934964
Doc 17	0.002329175898690642
Doc 18	0.00663611875824933
Doc 19	0.009067043341466102
Doc 20	0.017102597673059303
Doc 21	0.011903765982067665
Doc 22	0.007346363415240385

Abstract

Docs	Doc 2
Doc 0	0
Doc 1	0
Doc 2	1
Doc 3	0.1720370802687955
Doc 4	0.030613307435746005
Doc 5	0
Doc 6	0
Doc 7	0.11598777559654597
Doc 8	0.00914686577268861
Doc 9	0.007499992675791978
Doc 10	0
Doc 11	0
Doc 12	0
Doc 13	0
Doc 14	0.007740695138863195
Doc 15	0.039573325330024826
Doc 16	0.07462432962262808
Doc 17	0.012859808591210989
Doc 18	0
Doc 19	0
Doc 20	0.03343689557015695
Doc 21	0
Doc 22	0

Fig. 9. Cosine similarity results between biomedical documents

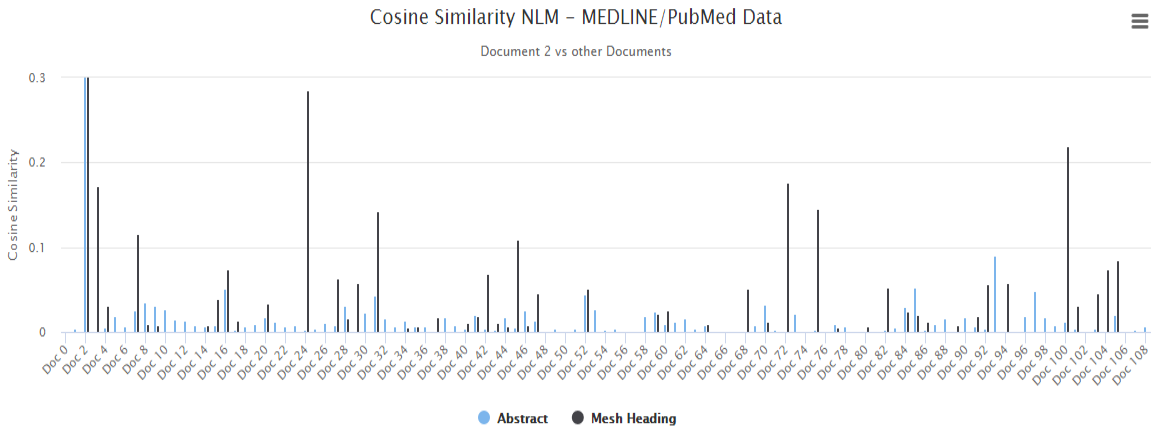


Fig. 10. Visualization of comparison of cosine similarity result between documents

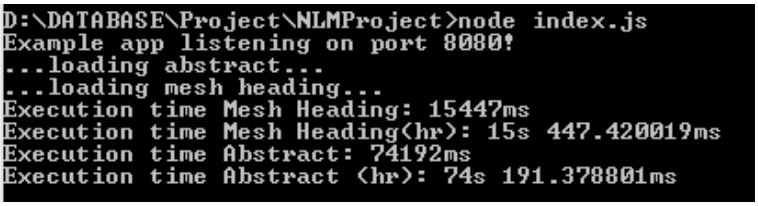


Fig. 11. Execution time of document similarity application

Both visualizations of the calculation similarity result depicted in Figure 9 and Figure 10, known as similarity identification results, make it easier to present and understand the comparison result. This identification similarity result is shown in the PubMed application. In addition, this result is produced by the parallel computing engine in the PubMed application that reduced the processing time and quickly detected the relationship and relevance between large-scale biomedical documents.

Meanwhile, Figure 11 is shown the execution time of the similarity engine application. The execution time of mesh heading is 15.447 seconds, and the timely execution of abstract is 74.191 seconds. The execution time of mesh heading is higher than abstract because abstract contains more words than mesh heading.

Documents similarity identification application has successfully identified the similarity between large-scale documents of the PubMed documents known as biomedical documents. The implemented cosine similarity and parallel computing as the document similarity engine is executed the documents faster. The execution time of mesh heading is 15.447 seconds, and the timely execution of abstract is 74.191 seconds. Based on the results, the mesh heading runtime is higher than the abstract because the abstract contains more words than the mesh heading. In addition, using the abstract and mesh heading tag can represent the similarity between documents. The result is shown that the cosine similarity of the mesh heading texts is higher than the mesh abstract text.

IV. Conclusion

The documents similarity identification application has successfully identified the similarity between large-scale documents of the PubMed documents known as biomedical documents. This study implemented cosine similarity and parallel computing as the document similarity engine that executed the documents faster. The execution time of mesh heading is 15.447 seconds, and the timely execution of abstract is 74.191 seconds. The mesh heading runtime is higher than the abstract because the abstract contains more words than the mesh heading. Therefore, using the abstract and mesh heading tag can represent the similarity between documents—the result is shown that the cosine similarity of the mesh heading texts is higher than the mesh abstract text. Besides, the results showed that the relevance between mesh heading text in the PubMed documents is higher than the relevance of the abstract text in the PubMed documents. On the other hand, the number of words and terms in the abstract can affect the percentage of text similarity results. In the future, this mesh heading and abstract tag can be used for the next data processing, such as classification or clustering datasets.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no known conflict of financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

Reprints and permission information are available at http://journal2.um.ac.id/index.php/keds.

Publisher's Note: Department of Electrical Engineering - Universitas Negeri Malang remains neutral with regard to jurisdictional claims and institutional affiliations.

References

[1] H. Hassani, C. Beneki, S. Unger, and M. T. Mazinani, "Text Mining in Big Data Analytics," *Big Data Cogn. Comput.*, vol. 4, pp. 1–34, 2020.

[2] R. Islamaj *et al.*, "PubMed Text Similarity Model and its application to curation efforts in the Conserved Domain Database," *Database*, vol. 1, pp. 1–13, 2019.

[3] S. F. Wamba, A. Gunasekaran, S. Akter, S. J. Ren, R. Dubey, and S. J. Childe, "Big data analytics and firm performance: Effects of dynamic capabilities," *J. Bus. Res.*, vol. 70, pp. 356–365, 2016.

[4] M. Wibowo, F. Noviyanto, S. Sulaiman, and S. M. Shamsuddin, "Machine Learning Technique For Enhancing Classification Performance In Data Summarization Using Rough Set And Genetic Algorithm," *Int. J. Sci. Technol. Res.*, vol. 8, no. 10, pp. 1108–1119, 2019.

[5] R. M. Packiam and V. S. J. Prakash, "An empirical study on text analytics in big data," 2016.

[6] M. Erritali, A. Beni-hssane, M. Birjali, and Y. Madani, "An Approach of Semantic Similarity Measure between Documents Based on Big Data," *Int. J. Electr. Comput. Eng.*, vol. 6, no. October 2017, pp. 2454–2463, 2016.

[7] L. A. Rahim, K. Mohan, K. Id, and S. Bahattacharjee, "Framework for parallelisation on big data," PlosOne 14(5), pp. 1–19, 2019.

[8] B. Parhami, "Parallel Processing with Big Data," pp. 1–7, 2018.

[9] R. Darmawan, R. S. Wahono, "Hybrid Keyword Extraction Algorithm and Cosine Similarity for Improving Sentences Cohesion in Text Summarization," *J. Intell. Syst.*, vol. 1, no. 2, pp. 109–114, 2015.

[10] S. W. Irianandа, M. A. Muslim, and H. S. Dachlan, "Identifikasi Kemiripan Teks Menggunakan Class Indexing Based dan Cosine Similarity Untuk Klasifikasi Dokumen Pengaduan," *Matics*, vol. 10, no. 2, p. 30, 2019.

[11] D. A. R. Ariantini, A. S. M. Lumenta, and A. Jacobus, "Pengukuran Kemiripan Dokumen Teks Bahasa Indonesia Menggunakan Metode Cosine Similarity," *J. Tek. Inform.*, vol. 9, no. 1, pp. 1–8, 2016.

[12] M. Z. Naf'an, A. Burhanuddin, and A. Riyani, "Penerapan Cosine Similarity dan Pembobotan TF-IDF untuk Mendeteksi Kemiripan Dokumen," *J. Linguist. Komputasional*, vol. 2, no. 1, pp. 23–27, 2019.

[13] J. Wang and Y. Dong, "Measurement of text similarity: A survey," *Inf.*, vol. 11, no. 9, pp. 1–17, 2020.

[14] D. Kurniadi, S. F. C. Haviana, and A. Novianto, "Implementasi Algoritma Cosine Similarity pada sistem arsip dokumen di Universitas Islam Sultan Agung," *J. Transform.*, vol. 17, no. 2, p. 124, 2020.

[15] D. Gunawan, C. A. Sembiring, and M. A. Budiman, "The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents," *J. Phys. Conf. Ser.*, vol. 978, no. 1, 2018.

[16] J. Bian, M. Amin, S. Jonnalagadda, G. Luo, and G. Del, "Automatic identification of high impact articles in PubMed to support clinical decision making," *J. Biomed. Inform.*, vol. 73, pp. 95–103, 2017.

[17] C. W. Halladay, T. A. Trikalinos, I. T. Schmid, C. H. Schmid, and I. J. Dahabreh, "Using data sources beyond PubMed has a modest impact on the results of systematic reviews of therapeutic interventions," in *Journal of Clinical Epidemiology*, 2015, vol. 68, no. 9, pp. 1076–1084.

[18] K. Z. Vardakas, G. Tsopanakis, A. Poulopoulou, and M. E. Falagas, "An analysis of factors contributing to PubMed's growth," *J. Informetr.*, vol. 9, no. 3, pp. 592–617, 2015.

[19] MongoDB, "MongoDB," 2017.

[20] P. dwi Nurfadila, A. P. Wibawa, I. A. E. Zaeni, and A. Nafalski, "Journal Classification Using Cosine Similarity Method on Title and Abstract with Frequency-Based Stopword Removal ," *Int. J. Artif. Intell. Res.*, vol. 3, no. 2, 2019.

[21] N. Ghasemi and S. Momtazi, "Neural text similarity of user reviews for improving collaborative filtering recommender systems," *Electron. Commer. Res. Appl.*, vol. 45, no. October 2019, p. 101019, 2021.

[22] M. Wibowo, S. Sulaiman, S. Mariyam, and H. Hashim, "Mobile Analytics Database Summarization Using Rough Set," *Int. J. Innov. Comput.*, vol. 7, no. 2, pp. 6–12, 2017.