Automatic keyword assignment system for medical research articles using nearest-neighbor searches

FATİH DİLMAÇ

ADİL ALPKOÇAK

Automatic keyword assignment system for medical research articles using nearest-neighbor searches

Fatih DİLMAÇ[1,*] **, Adil ALPKOÇAK**[1,2]

[1]Department of Computer Engineering, Dokuz Eylül University, İzmir, Turkey
[2]Department of Computer Engineering, Bakırçay University, İzmir, Turkey

Abstract: Assigning accurate keywords to research articles is increasingly important concern. Keywords should be selected meticulously to describe the article well since keywords play an important role in matching readers with research articles in order to reach a bigger audience. So, improper selection of keywords may result in less attraction to readers which results in degradation in its audience. Hence, we designed and developed an automatic keyword assignment system (AKAS) for research articles based on k-nearest neighbor (*k*-NN) and threshold-nearest neighbor (*t*-NN) accompanied with information retrieval systems (IRS), which is a corpus-based method by utilizing IRS using the Medline dataset in PubMed. First, AKAS accepts an abstract of the research article or a particular text as a query to the IRS. Next, the IRS returns a ranked list of articles to the given query. Then, we selected a set of documents from this list using two different methods, which are *k*-NN and *t*-NN representing the first *k* documents and documents whose similarity is greater than the threshold value of *t*, respectively. To evaluate our proposed system, we conducted a set of experiments on a selected subset of 458,594 PubMed articles. Then, we performed an experiment to observe the performance of AKAS results by comparing with the original keywords assigned by authors. The results we obtained showed that our system suggests keywords more than 55% match in terms of F-score. We presented both methods we used and results of experiments, in detail.

Key words: Automatic keyword assignment, information retrieval, k-nearest neighbors, t-nearest neighbors, PubMed

1. Introduction

During the last decade, technology and the Internet is developing widely and quickly. That leads us to work with great and massive data daily, and consequently it produces millions of unstructured data every millisecond. Hence, we need to control the flow and preservation of data in a way that makes it easy to access and retrieve. Instead of reading the whole content of the these documents, he/she can just look at keywords that represent the topics of that document.

The keywords are generally a few words or word phrases that best describe the context of the articles [1]. Regardless of the contents of the articles, researchers can only search the keywords or the topics of the articles, with the help of the search engines to get related articles. Therefore, keywords should be chosen very carefully. However, assigning keywords is an important and difficult task. Performing this process manually is costly and may cause improper keyword assignment. If we take care and focus on assigning keywords to research articles, then it is possible to access these studies more efficiently.

---

*Correspondence: fatih.dilmac@ceng.deu.edu.tr

To solve these problems text mining researchers are spending more effort to identify and assign keywords to research articles automatically with the state of art techniques [2, 3]. Many unsupervised techniques have been used for keyword extraction [4–6]. These approaches have accomplished satisfactory results and are widely used in literature. However, working with keywords still has some difficulties, especially if we have to deal with the keyword assignment process. This is happening because keywords sometimes are assigned by authors, which will make the assignment process subjective to the author's opinion. Moreover, keyword assignment is a multilabel problem in which more than one keyword is assigned to a single text that is directly relevant to the topic at hand [7–9]. The motivation with any multilabel problem is to find the most relevant and sufficient number of labels, or keywords in our case, that will maximize the performance of classification systems. Additionally, the major challenge in this problem is to deal with two problems: label dependency and the huge set of labels. For that reason, we need a way that correctly predicts a limited set of interrelated keywords. Therefore, we proposed an automatic keyword assignment approach to a given article, which is based on the nearest-neighbors algorithm.

Our motivation is to prove the effectiveness of assigning keywords to medical articles automatically. Despite of the existence of many studies on keyword extraction, very few studies have addressed automatic keyword suggestion systems. In addition, the lack of such studies on the Medline data set is one of the most important reasons that push us to work in the medical domain. Replacing the subjective and manual keywords assignment of the research articles with an automatic-based method is another reason behind the working on keyword suggestion system.

One of our most important contributions in this study is the development of this application by combining information retrieval systems and text mining techniques for the first time on scholarly articles. Additionally, the keyword assignment process can be carried out utilizing a variety of methods, including machine learning, statistical analysis, and so on. The datasets employed in these strategies are either relatively small or balanced. However, the data we collected from the Pubmed data set in this study includes an extensive number of classes and unbalanced. According to the [8–10], machine learning algorithms and deep learning approaches cannot deal with the former mentioned problems. As a result, we decided to use information retrieval techniques to handle such challenges. Another contribution of our research to the literature is a threshold-based nearest neighbor search (t-NN) technique. This method suggests keywords based on the similarity ratio of the papers. To the best of our knowledge, this is the first automated keyword suggestion system for scholarly articles written in the biomedical field.

The paper is organized in six main sections as follows: In Section 2 we present a literature review about keyword extraction using text mining. In Section 3, we explained the methods, the tools and all the phases involved during the system development, in detail. We also presented how we collected the Medline dataset, which we use as a corpus in our system. In Section 4, we explained the experiments and the results in detail. In Section 5, we discussed the advantages and drawbacks of our system. In Section 6, the conclusion of this paper is given by summarizing the overall results and the contribution accompanied with some ideas as future work.

2. Related work

In this section, we represent how researches in the literature approach the work on keyword extraction techniques in text mining. We can categorize approaches commonly used in the literature for keyword extraction into four main groups: machine learning, statistical, graph-based, and linguistic approaches. Moreover, we can categorize machine learning approaches [11] into three subgroups: supervised, unsupervised [12], and semisupervised

approach [13].

2.1. Statistical approaches

In this approach, the researchers do not depend on the semantic representation of the text to extract keywords or key-phrase, instead they consider only cooccurrence of terms such as term frequency (TF) and Term frequency inverse document frequency (TF-IDF). Matsuo et al. developed a new approach that can be used on a single document instead of the entire corpus. They used an approach that involves calculating the frequency of each term in each document and then selecting only the most often occurring terms within a given threshold. So that if two terms appear together (Bi-gram) with higher chi square X^2 then it will be chosen to represent a keyword in that document [14]. While Ruch et al. evaluated the impact of the argumentative structures of scientific articles to improve the effectiveness of the classification process. Their algorithm combines linguistically-motivated and information retrieval methods [15]. Moreover, Pay et al. created an ensemble method to extract keywords from a single document. For ensembling, they used TextRank, RAKE, and TAKE approaches [16]. Singhal et al. used Renyi entropy to construct a statistical approach-based domain-independent keyword extraction model. This model ranks words based on word frequency and word cooccurrence probability distribution[17]. Finally, Onan et al. tried to construct a new ensemble keyword extraction model by combining existing statistical keyword extraction methods: the most frequent measure-based keyword extraction, term frequency and inverse-sentence frequency, cooccurrence statistical information, eccentricity-based and TextRank algorithms [18].

2.2. Linguistics approaches

These approaches use natural language processing techniques to tag the words of the language and divide the sentences into words and phrases. Keyword extraction is performed depending on analyzing lexical, syntactic, semantic and discourse of corpus [19]. Zhang et al. collected Chinese documents from the "Information Center for Social Sciences of RUC" database. They used the set-tag tools of the NLP library for POS tagging which is automatically processing the keywords of each document. Consequently, their experimental results indicate that the conditional random fields (CRF) model performs better than other machine learning algorithms, such as SVM, MLR model, etc., for keyword extraction [10]. In another study researchers tried to show how rapid automatic keyword extraction algorithm (RAKE) is efficient for keyword extraction. RAKE is an unsupervised learning and domain-independent technique that specifies keywords from a single document. Eventually, they found that RAKE is faster than other algorithms by comparing RAKE with existing keyword extraction algorithms such as TextRank and unsupervised learning techniques [20].

2.3. Machine learning approach

Machine learning approaches can be classified into supervised or unsupervised learning. However, previous studies on keyword extraction get better results from supervised learning techniques. Hence, they need manual annotations to learn from the dataset which is an exhaustive process. In this approach researchers have used one of the following machine learning algorithms: naïve Bayes, SVM (support vector machines), C4.5, etc. [21, 22]. Furthermore, these models are mostly dependent on the domain. A system needs to relearn and establish the model every time when a domain changes. Another proposed a novel system called ACRI (associative classifier with cooccurring items) which automates the classification of MEDLINE documents to MeSH keywords. Their proposed system was modified to represent a multilabel classification problem. Their results show the superiority

of the proposed methods based on cooccurrence of words in an article over nonrecurrent-based associative classification [23], while, Kamal Sarkar proposed key phrase extraction method which consists of three primary components: document preprocessing, noun phrase identification, and key phrase identification. The method is based on the naive Bayesian learning that exploits a number of domain-specific features to boost up the key phrase extraction performance in the medical domain. However they did not use the Medline dataset. The proposed method has been compared to a popular key phrase extraction algorithm, called Kea [3]. Moreover, HaCohen-Kerner and his friends proposed a model for key phrase extraction based on supervised machine learning and combinations of the baseline methods [24]. Finally, Turney proposed a key phrase extraction algorithm based on supervised learning. A document is treated as a set of phrases, which must be classified as either positive or negative examples of key phrases based on examination of their features. The algorithm considers the location of the first occurrence of a phrase in a document, the frequency with which a phrase occurs within a document, whether the phrase is a proper noun and whether or not a phrase matches a human-generated key phrase [25].

2.4. Graph based approaches

Graph based techniques give a useful insight about the relationships between words in documents. Each word can be represented as vertices or nodes and a link between them can show the relationship appropriately [26]. This representation can graphically make abstracted information about the documents and easily traceable. Keywords on the other hand can be represented as graph [11, 27]. Another proposed a graph-based framework is Topical PageRank (TPR) on a word graph to measure word importance with respect to different topics [7]. Xiong et al. proposed Semantic Clustering TextRank (SCTR), a semantic clustering news keyword extraction algorithm based on TextRank. They generate word vectors by using the BERT model. Following that, word graphs are constructed and keyword extraction is performed [28]. Finally, another research paper proposed two graph-based approaches: supervised and unsupervised, for the cross-lingual keyword extraction to be used as extractive summarization of text documents [21]. The performance of their supervised graphical model is improved by conducting a classification model depending on the graphical representation of the document features [29].

3. Methodology

In this section, we presented details of the automated keyword assignment system (AKAS) that is built upon an information retrieval system (IRS). Firstly, we presented the whole system in an architectural point of view describing the major components and main phases of the system. Then, it provides a detailed formal definition of the system. Figure 1 shows a general view of the system we propose, which includes four basic components: the user interface, IR system, database, and ranking subsystem. In this section, we explain each component in detail flowing from bottom to top:

3.1. Database module

The AKAS uses the Medline dataset from PubMed, which is a free-of-charge database of the US National Institute of Health where a number of biomedical publications are indexed together. Our dataset includes abstracts and other information of research articles from PubMed, which includes around 32 million articles and the size of the whole dataset is approximately 170 GB PubMed. This dataset is stored in 1015 files in XML each containing 1000 articles. First, we downloaded and imported the entire dataset into our database
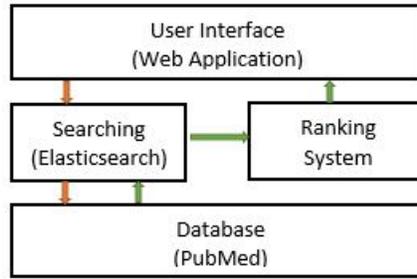
Figure 1. Architectural view of the automated keyword assignment system (AKAS).

running MongoDB [30, 31] which we installed on our own server. Then we converted the stored data articles into JSON format and saved them into our local MongoDB database. Since the PubMed database is continuously updated, the database module checks for new XML files from the PubMed server periodically on a daily basis to keep our database up-to-date. In this study, a subset of the PubMed articles published in the year 2018 were used and it includes 458,594 articles in total. We divided this dataset into a training and testing set, which includes 450,000 and 3594 articles, respectively. Table 1 contains the detailed statistics of the PubMed dataset that are stored in our database. The dataset is open to researchers for further investigations on the topic (http://demir.cs.deu.edu.tr).

Table 1. Basic statistics about the dataset used in this study.

Descriptions	Statistics
Total number of documents	458,594
Number of documents in sentences / tokens / characters	4,543,908 / 101,990,973 /667,981,517
Total Number of unique keywords	25,666
Maximum number of keywords / minimum / average	58 /2 / 17
Training data unique keywords / test data unique keywords	25,641 / 7,733
Max document size in sentence / tokens / characters	85 / 1,482 / 9,825
Min document size in sentence / tokens / characters	1 / 3 / 12
Average document size in sentence / tokens / characters	10 /147 / 222

3.1.1. Word feature representation

In this subsection we present more details about the preprocessing and the text representation approaches that we performed before experimentations. We represent each word in the PubMed abstract text as one of the following feature representation techniques: word to vector, document to vector, and named entity extraction. Each method is assigned and applied to explore its impact on the performance of our IRS model. As data preprocessing, we first removed punctuation chars, digits, stop-words and then applied lemmatization to each word. Finally, we saved the preprocessed abstract field as cleaned-abstract.

3.1.2. Word to wector representation

We used fastText [32] to create word vectors with each word in the `cleaned_abstract` field. FastText is a library for creating word embedding designed by Facebook's AI Research lab. Each word w in our dataset has

a vector that consist of a float point 1×100 dimension, such that $\vec{w} = [\ f0,\ f1,\ ...,\ f99\ ]$. Hence, a lookup table then is created to represent each word w with its vector. Additionally, we created two new fields by taking the sum and average of the documents' vectors for each document, namely $\overrightarrow{(vector\_sum)}$ and $\overrightarrow{(vector\_avg)}$. For creating the word vectors, we followed the same steps in study [8]. As a formal definition to clarify how the vectors are created, let us assume that our document collection D = {d1, d2, d3, ..., dM} and each di = { *abstract, cleaned_abstract, vector_sum, vector_average, vector_doc2vec, ner_text, keywords* } data fields, where $d \in D$ and ℓ is number of words in the *cleaned_abstract* field for di. Each word is represented by 1×100-dimension vector as $\vec{w}$ that is ready to further create *vector_sum* and *vector_average* as it is declared in Equations 1 and 2.

$$vector\_sum[di] = \sum_{i=0}^{l}(vector\_sum + = \vec{w_i}) \tag{1}$$

$$vector\_avg[di] = \frac{1}{l} * \sum_{i=0}^{l}(vector\_avg + = \vec{w_i}) \tag{2}$$

3.1.3. Document to vector representation

In this data field, we used the *doc2vec* library of the gensim [33] module. Each document now is represented as 1×100 vector and saved as a new field with the name $\overrightarrow{vector\_doc2vec}$.

3.1.4. Name entity representation

Finally, we used 3 pretrained models of the scispacy [34] library that are created for named entity recognition (NER) in the biomedical domain: *en_ner_bionlp13cg_md, en_ner_bc5cdr_md*, and *en_ner_jnlpba_md*. Each NER pretrained model extracts different medical entities such as amino_acid, anatomical_system, cancer, cell and so on.

3.2. Search module

The search module takes the query from the user through the UI module and finds the most similar documents to the given query in a ranked order. It is worth mentioning that the search module weights those documents by calculating the similarity scores of each document according to the entered query. Finally, it returns a list of the resulting documents according to its similarity score in a descending order.

3.2.1. Information retrieval system

We used Elasticsearch [35] as an information retrieval system (IRS). Elasticsearch is a file based and distributed search engine which is not similar to relational databases. People have two important reasons to use Elasticsearch as their retrieval systems, especially in text search fields: its strong indexing structure and its internal inverted indexing structure. Elasticsearch uses a Boolean retrieval function to get all the relevant documents which contain query terms. Then it calculates the term frequency-inverse document frequency, TF×IDF, for all the terms in the retrieved documents. TF×IDF method weights a term to describe its importance in an individual document within a corpus. Let us assume that l, t and d denote number of words in document *d*, a term and document respectively such that $d \in D$, where t appears in n of N documents in D where N is number of

documents in corpus. The TF×IDF function is described in the following form:

$$TF \times IDF(t, d, n, N) = TF(t, d) \times IDF(n, N) \tag{3}$$

$$TF(t,d) = \sum_{t \ \in \ d}^{l} (tf = tf + 1 \ if \ t \ in \ document \ (d) \tag{4}$$

$$IDF(n, N) = log(\frac{n}{N}). \tag{5}$$

Finally, it creates vectors for all document in the result-set RS, then calculates the cosine similarity between query q and each document di where F is the overall feature set and f is the ith feature

$$Cos(q,di) = \frac{\sum_{f \ \in \ F}(q_f \times di_f)}{\sqrt{\sum_{f \ \in \ F}(q_f^2)} \times \sqrt{\sum_{f \ \in \ F}(di_f^2)}}. \tag{6}$$

3.3. Ranking component

This component depends on ranking the candidate keywords of the obtained result set of RS. It performs four main processes: retrieval of the result set using k-NN method, retrieval of the result set using t-NN method, and it reranks the result set based on another two approaches: frequency-based ranking (FBR) and score-based ranking (SBR). These two approaches are used to retrieve the relevant RS independently, i.e. not at the same time. After that for each of the methods we used two ranking approaches, frequency-based ranking (FBS) and score-based ranking (SBR), to control the appropriate top n candidate keywords suggested to the users. After all, the users can select suggested keywords.

3.3.1. k-NN method

We used the k-NN method to control the RS of the IRS by considering the top k of the results. Primarily, we used the abstract of the PubMed article as an input query to the IRS. Then, the IRS returns top k similar articles accordingly. After that, we applied FBR or SBR sub approaches on the retrieved RS to rank the keywords list. In this list we select the top n, number of keywords, to the new article. Finally, we obtain the optimal k and n parameters in terms of the highest F-scores.

3.3.2. t-NN method

In the t-NN method we control the amount of RS according to a particular threshold (t) of the document's similarity scores. Same as k-NN, we first feed the abstract of the PubMed article to the IRS. Then IRS returns the top RS that their similarity scores are higher than a threshold t. Then we apply FBR and SBR approaches to the returned result set and accomplish the keywords list. In this list we suggest also the top n, number of keywords, keywords to the new article. The rank order field of Table 2 shows m number of RS retrieved as a result of applying a particular query to the search component. For example, in k-NN, let k be a list of values: 2, 4, 6, and 10. That means for every query Q the result-set RS of IRS can be represented as the top 2, 4, 6 or 10 documents respectively. Let 6 be the optimal k value which provides us with the maximum F-score. Accordingly, we are considering the top 6 RS documents to be the optimal option.

On the other hand, Table 2 also describes how the t-NN method works. For example, under the same query as in the example of k-NN approach, m number of RS documents were retrieved. Each document is represented with its related keywords, that is represented by the keywords field in Table 2, and similarity scores, which also is represented by the similarity score field of Table 2. Then if we set a threshold $t = 80$, that means only the keywords list of the RS documents with similarity scores greater than 80 will be taken into account. Taking into consideration that each method is executed individually not at the same time.

Table 2. k-NN and t-NN methods on the IR results.

Result set (sorted by similarity score)		
Documents	**Keywords**	**Similarity score**
1	B, C, F, G	500
2	B, C, G	400
3	A, D	100
4	A, E, D	90
5	A, E	85
6	A, E, C, D	80
7	G, H	55
8	X, Y	40
m	A, B	10

3.3.3. Frequency-based ranking (FBR)

Both k-NN and t-NN methods use this approach. FBR approach depends mainly on calculating the frequency, i.e. normal count, of each keyword that exists in RS documents and retrieved as a result of both: the k-NN or t-NN aforementioned techniques. After that, the system arranges the frequencies of all the distinct keywords in a descending order. Accordingly, we depend on the top n keywords that have the maximum frequency score. For more elaboration about how the scores are calculated, let us assume that number of documents in $RS = m$, which IRS returns, and x be the number of keywords k in document di,

$$RS = \{d1, d2, d3, ..., dm\}$$

$$di = \{k1, k2, k3, ..., kx\}$$

.

Each document in RS has a list of relevant keywords assigned by its authors. We used FBR and SBR algorithms to rank candidate keywords in the result set RS. RS$_f$ is the ranked keyword list obtained by FBR method and can be calculated as follows:

$$RS_f = \sum_{i=1}^{m} \sum_{j=0}^{x} \begin{cases} RS_f[k_j]+ = 1 & if\ k_j\ in\ RS_f \\ RS_f[k_j] = 1 & otherwise \end{cases}. \tag{7}$$

3.3.4. Score-based Ranking (SBR)

This technique/approach is also considered in both k-NN and t-NN main methods. SBR calculates keywords score by multiplying the keywords of the document with its related similarity score, i.e. for each keyword

belonging to the documents we obtained with the IRS. RS_s is the ranked keyword list that is obtained by SBR method and can be calculated as follows:

$$RS_s = \sum_{i=1}^{m} \sum_{j=0}^{x} \begin{cases} RS_s[k_j] + = Score^{kw_j} & if\ kw_j\ in\ RS_s \\ RS_s[k_j] = Score^{kw_j} & otherwise \end{cases}. \tag{8}$$

For example, suppose that the IRS retrieved m relevant set of RS, as it is shown in Table 2, according to k-NN or t-NN main techniques: let B, C, F, G be the retrieved keywords of document1 as it is shown in Table 2. In FBR, only the top 6 of the RS is taken into account –as a result when $k = 6$– then the distinct keywords A, B, C, D, E, F, and G will be considered in Table 3. Each keyword will be counted individually to best describe the frequency of each keyword. Then the top n keywords will be retrieved. So, if $n = 4$ then the 4 keywords, A, C, D, and E, respectively, will be suggested as seen in Table 3. Similarly, in SBR technique, each keyword, as in Table 2, in document1 now will be multiplied by its similarity weight 500. The resultant keyword list now will be represented as 500B, 500C, 500F, and 500G, respectively. The same process will be repeated for all the documents in the RS. After that we sum all the weighted distinct keywords and if $n = 4$ then the top 4 keywords, C, B, G, and F respectively, with the highest scores will be suggested as seen in Table 3. As can be seen, both techniques produce different keywords.

Table 3. FBR and SBR results on k-NN and n-NN methods.

Keywords	Frequency	FBR	Similarity score	SBR
A	4	1	100 + 90 + 85 + 80 = **355**	5
C	3	2	500 + 400 + 80 = **980**	1
D	3	3	100 + 90 + 80 = **270**	6
E	3	4	90 + 85 + 80 = **255**	7
B	2	5	500 + 400 = **900**	2
G	2	6	500 + 400 = **900**	3
F	1	7	**500**	4

3.4. UI module

The final component of this application is the user interface (UI) component. We have developed a simple interface for the end users to ease the use of this application. We used the Flask [36, 37] web framework for developing the interface. This framework, which is commonly used among web-based software developers, is ideal to do something quickly and bring out certain results in time. One of the most important features of Flask is the MVC (model-view-controller) architectural design pattern that can be configured very easily. Figure 2 shows the main user interface of the application we developed. The input field shows the abstract that the user will use as a query. The query can be an abstract of a biomedical article or any text in accordance with the Medline data set. As a result, when a user clicks on the "Suggest Keywords" button, the UI application sends the query to the search component. In other words, the search component sends the query to the database and expects the relevant documents. Then it sends the RS to the ranking component. After that, the ranking component chooses either FBR or SBR techniques to send the relevant keywords and RS to the user interface.

Figure 2 describes one web page with a user query and a suggested list of keywords. Moreover, it also shows the retrieved similar documents accompanied with their details such as document id, abstract and similarity scores.

Figure 2. The user interface of the web application.

4. Experimentation and evaluation

In our experiments we used Elasticsearch which is a distributed NoSQL database and Apache Lucene [35] based search engine. As a result, we have completed our web application using the previously mentioned technologies and it is available on application link. To evaluate the quality and proficiency of our system, we used commonly used metrics of Jaccard similarity, F-measure, precision and recall @n rates to assess the results. Definitions of variables used in formulas: A (Author's Keywords) represents the set of keywords which are assigned by authors to articles in Medline dataset, S (Suggested Keywords) represents the set of keywords suggested by our algorithm, $|A \cap S|$ represents the number of the intersection of A and S. Finally, n represents the top of the result set. The commonly used formulas are as follows.

For the top n of ground truth and the suggested keywords, precision@n which is generally indicated by the letter P@n, is calculated as the ratio of the intersection between the ground-truth and the suggested keywords to the top n of the suggested samples.

$$P@n = \frac{|A \cap S|}{|n|} \tag{9}$$

For the top n of ground truth and the suggested keywords, recall@n which is also indicated by the letter R@n, is calculated by the ratio of the intersection between the ground truth and the suggested keywords to the

total number of the ground truth keywords.

$$R@n = \frac{|A \cap S|}{|A|} \tag{10}$$

In light of these definitions above, the F-measure is the harmonic mean of these values.

$$F - score = \frac{P \times R \times 2}{P + R} \tag{11}$$

Finally, the J represents Jaccard similarity and it is a statistical term used for measurement the similarity of sample sets. Assume that A (Author's Keywords) represents the set of keywords which are assigned by authors to articles in Medline dataset, S (Suggested Keywords) represents the set of keywords suggested by our algorithm, $|A \cap S|$ represents the number of the intersection and $|A \cup S|$ represents the number of the union of A and S. The commonly used formulas are as follows.

$$J = \frac{|A \cap S|}{|A \cup S|} \tag{12}$$

4.1. Experiments on k-NN

We conducted many experiments to find the optimal values of k (number of result set) and n (number of keywords) that are retrieved by IRS. Accordingly, we assign k= 3, 7, 11, 15, 19, and 23. We need to see which value of k will register the best performance of our IRS model. We basically considered two different ranking techniques FBR and SBR.

4.1.1. Parameter optimization for frequency- and score-based ranking (FBR and SBR)

In FBR technique, we aim to find the optimal n that will give us the best F measure score. Firstly, we set $n = 3$ to 29, to observe the optimal (k, n) pairs in terms of F-score. Secondly, for each k number of a particular query's result set, we calculate the frequency of keywords. Then we sorted them in a descending order according to their frequencies. Finally, we suggest top n keywords accordingly. After all, we calculate the F-score to evaluate our system. We represent the scores of F-score as shown in Figure 3. Similarly, in the score-based ranking technique, we set $n = 3$ to 39 to observe the optimal (k, n) pairs in terms of F-score. Then, for each k number of a particular query's result set, we calculate the score for each keyword with its document similarity. Then we sorted it in a descending order according to the similarity score. Finally, we suggest top n keywords. Eventually, we calculate the F-score to evaluate our system. We represent the scores of F-score as shown in the curves in Figure 3. The figure represents the F-scores of all the k values that are used in this experiment. For example, the value of $k = 15$ and the value of n= 15, F-score reaches its maximum.

4.2. Experiments on t-NN

We select different values of t, from t= 0.2 to 0.9, with setting different values of n in both: FBR and SBR. After that we observe which t value gives us the maximum F-score to take it into account.

4.2.1. Parameter optimization for frequency- and score-based ranking (FBR and SBR)

In this section, we have conducted a set of experiments to find the optimal values of the threshold t in the t-NN-frequency method FBR. Thus, we run experiments for different sets of n values (i.e. $n = 3$ to 29) to determine the optimal number of keywords. Then we selected the optimal t and n values from the obtained results according to the F-score measure similar to t-NN in Figure 4.
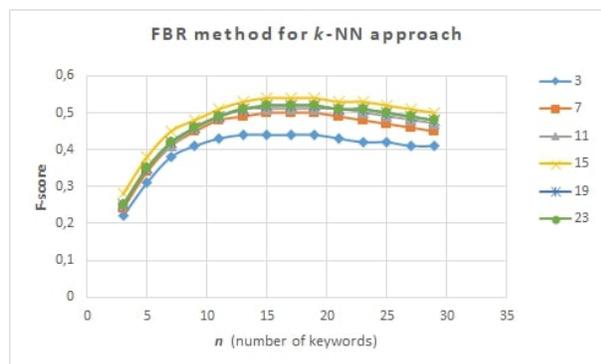

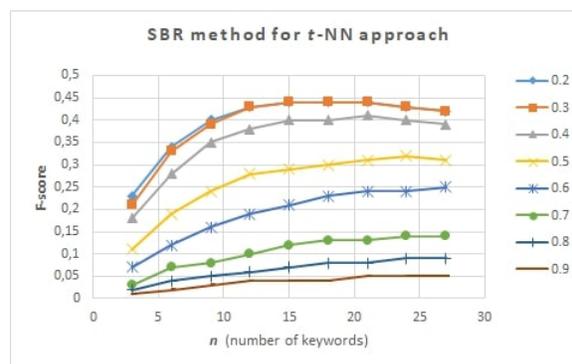**Figure 3**. FBR optimal value for k and n.   **Figure 4**. SBR optimal value for t and n.

In SBR technique, the process goes through steps that are similar to FBR. However, instead of finding a particular keyword frequency, the keywords of the articles are weighted by multiplying the similarity score of the article. For that reason, we selected the optimal t value from the results obtained according to the F-score as it is shown in Figure 4.

Furthermore, Figures 3 and 4 show that each colored curve is represented by a given k or t and its corresponding n values. For example, the blue line in Figure 4 indicates that the value of t is 0.2 and the value of n changes with respect to F-score. After that, we evaluate the system in terms of precision @ 1 to 10 on each examined data field;

Moreover, Elasticsearch's default scoring function works based on TF-IDF weighting technique. Hence, score values are greater than 1 specially when we start a query on text fields: such as abstract, cleaned_abstract or ner_text. Alternatively, in vector fields, such as vector_sum, vector_avg and vector_doc2vec, the result of utilizing the cosine similarity technique was a score that varied from 0 to 1. Therefore, we need to unify both value ranges to work in the same space. For that purpose, we combined the ranges using MIN-MAX normalization technique.

5. Results and discussion

In this study, we performed several experiments to determine the optimal values for the k, t, and n variables. As a result of these experiments, we evaluated the F-score values of (k,n) and (t,n) pairs. Then, as seen in Figure 3, when $k = 15$ and $n = 15$, the F-score reaches its maximum value. Therefore, we selected these optimal values for the k-nn method. After doing the same process in t-nn, the optimal values selected for (t,n) pair. Then, using these optimal values, we calculated precision and recall values for k-nn and t-nn methods using FBR and SBR ranking techniques. Then, we evaluate the proposed methods in terms of P@1, 3, 5, 7, and 10 on each examined data field. Consequently, we gained four precision tables and four recall tables. We only presented precision and recall scores in detail for the FBR technique of the k-nn method in Table 4 and Table 5, respectively.

As a result, instead of presenting all these tables in detail, we summarized the best precision and recall scores of both methods in Tables 6 and 7, respectively.

Table 4. Precision@n results of FBR method on k-NN.

Method/n	P@1	P@3	P@5	P@7	P@10
abstract	0.960	0.875	0.810	0.752	0.675
abstract_cleaned	0.957	0.873	0.804	0.749	0.675
ner_text	0.923	0.798	0.716	0.653	0.578
vector_doc2vec	0.949	0.814	0.729	0.660	0.584
vector_sum	0.829	0.602	0.483	0.411	0.344
vector_avg	0.829	0.602	0.483	0.411	0.344

Table 5. Recall@n results of FBR method on k-NN.

Method/n	R@5	R@10	R@15	R@20	R@40	R@60	R@80	R@100	R@140
abstract	0.222	0.371	0.472	0.542	0.687	0.745	0.775	0.795	0.824
abstract_cleaned	0.220	0.370	0.470	0.540	0.683	0.744	0.774	0.795	0.824
ner_text	0.196	0.317	0.397	0.455	0.580	0.635	0.664	0.686	0.722
vector_doc2vec	0.200	0.321	0.402	0.460	0.585	0.643	0.673	0.693	0.731
vector_sum	0.132	0.188	0.227	0.256	0.323	0.349	0.362	0.376	0.403
vector_avg	0.132	0.188	0.227	0.256	0.323	0.349	0.362	0.376	0.403

Table 6. Precision@n results of FBR and SBR approaches on k-NN and t-NN methods.

Methods	k-NN		t-NN	
Approaches	FBR	SBR	FBR	SBR
Precision@n	n = 1			
abstract	0.960	**0.961**	0.890	0.885
abstract cleaned	0.957	0.955	0.875	0.875
ner text	0.923	0.925	0.685	0.685
vector doc2vec	0.949	0.949	0.890	0.890
vector sum	0.829	0.829	0.860	0.860
vector avg	0.829	0.829	0.860	0.860

Table 7. Recall@n results of FBR and SBR approaches on k-NN and t-NN methods.

Methods	k-NN		t-NN	
Approaches	FBR	SBR	FBR	SBR
Recall@n	n = 140			
abstract	**0.824**	**0.824**	0.780	0.790
abstract cleaned	**0.824**	**0.824**	0.780	0.780
ner text	0.722	0.722	0.600	0.610
vector doc2vec	0.731	0.731	0.770	0.780
vector sum	0.403	0.403	0.530	0.530
vector avg	0.403	0.403	0.530	0.530

From Table 4, we observed that the best precision score is obtained when the original abstract is used. That means Elasticsearch cannot perform well when some changes, such as preprocessing or NE extraction, are performed. It needs more data information to retrieve the relevant RS. While we lose information or data sequences when we do NE extraction and preprocessing, respectively. On the other hand, we noticed that document-to-vector technique works much better than word-to-vector technique. However, document-to-vector is a good vector representation that can be used as a dimensionality reduction technique.

In Table 5, we can observe that as the number of proposed keywords n increases, the recall score also increases. This indicates that as the suggested number of keywords grows, more keywords related to the relevant domain are recommended. This allows system users to choose the most relevant terms from a list of n keywords.

Table 6 shows that the best precision score is obtained with the original abstract and when k-NN and SBR techniques are addressed. As a result, Elasticsearch cannot work efficiently when certain changes, such as NER or preprocessing, are made. For the objective of retrieving the relevant RS, more data is required. While NER and preprocessing both result in information or data sequence losses, respectively.

Table 7 shows that when n is increased the recall is slightly increased as well. However, in Figure 3, at $k = 3$ the precision is getting much larger than the recall which means that our model at that point is more specific to retrieve the correct keywords. Moreover, we observed that the maximum recall score is obtained when $n = 140$ on abstract and abstract_cleaned fields for both techniques in k-NN. However, choosing an abstract_cleaned field can increase the model performance due to dimensionality reduction. Though the increase may still be continuing as n is increasing.

Additionally, we calculate Jaccard similarity (J) scores of the k-NN and t-NN methods. For k-NN method the best score of J is 36% obtained using the optimal ($k = 15, n = 15$) values in Figure 3. The highest J score for t-NN is 20% when the optimal ($t = 0.3, n = 15$) values selected in Figure 4 in terms of highest F-Score. According to the Jaccard similarity, the higher the similarity ratio, the more similar the two sets will be. Since the number of keywords suggested in our experiments ($n = 15$) is higher than the number of words assigned by the author, the jaccard similarity score is low. This metric is not used much in the literature. For example, assume that the author has assigned a total of 3 keywords, *a, b, c.* and our system suggests 15 keywords as *a, b, c...j.* In this case; although we recommended all the keywords assigned by the author correctly, Jaccard similarity ($3/15 = 0.2$) is still very low.

Table 8 shows the results of the studies, as well as the datasets and models that were employed. We can observe that previous research's datasets were often small and balanced. Regardless, the model we presented outperformed those in terms of precision, despite the fact that the dataset we utilized was both enormous and unbalanced.

Table 8. Comparison of our method with previous studies.

Studies	Datasets	Size	Model	Precision	Recall	F-score
[2]	ACM, Reuters and Web Doc	350	SVM	0.51	0.86	0.63
[3]	Online medical journals	75	Naive Bayesian	n = 5: 0.47 n = 10: 0.28	-	-
[14]	Technical papers	20	Word cooccurrence	0.42	0.46	-
[16]	Journal papers	2000	Ensemble method	0.46	0.5	0.48
[19]	Inspect dataset	2000	GenEx	0.25	0.52	0.34
[20]	Inspect dataset	2000	Graph based	0.33	0.41	0.37
[24]	Academic papers	161	J48	n = 5: 0.55 n = 15: 0.28	-	-
[25]	Journal articles, Email messages	656	GenEx	0.29	-	-
[28]	Chinese news	500	SCTR	n = 5: 0.7	-	-
Our method	PubMed	**458,594**	IR	n = 1: 0.96 n = 5: 0.81 n = 10: 0.67	n = 5: 0.22 n = 10: 0.37	0.55

Additionally, we did several experiments and discovered that the values for the *k-nn* technique ($k = 15$ and $n = 15$) and for the *t-nn* method ($t = 0.3$ and $n = 15$) were optimal. These values were then used to evaluate precision and recall metrics. However, most of the research in the literature used the value of *n* at 5 and 10. Therefore, we presented the results of the *n* value at 5 and 10 in the table to make comparisons.

There is also a conflict relation between recall and precision, as seen in Tables 6 and 7. The key explanation here is, as the number of *n* increases, the recall increases while the precision goes down.

We also noticed that in most of our experiments, the model performed better on a raw data. In addition, we found that the *k*-NN approach gave better results than the *t*-NN approach. In this case, it is more advantageous to use *k*-NN because the larger the RS size, the more time the ranking algorithm takes.

Additionally, the specificity or the precision of retrieving the correct set of keywords depends on assigning *k* instead of *t*. Because assigning *k* will ensure limiting the results to the most relevant keywords while *t* will retrieve a wide range that depends on the similarity score. For example, the IR system return RS= (d1, s:500), (d2, s:400), (d3, s:300), (d4, s:250), (d5, s:200), (d6, s:100) which has 6 documents with their similarity scores. When we set $k = 2$ then our system will consider only the first two most similar documents. On the other hand, there is no guarantee how many documents will be retrieved, it totally depends on similarity scores which are calculated based on a given query. In general, the efficiency of our system depends mainly on the proper and the correct number of the retrieved keywords. We also considered keyword assignment as a multilabel classification problem. Multilabel classification is a real-world challenging problem specially when it deals with a huge number of labels set [38, 39]. Consequently, in many machine and deep learning applications, using hundreds or even thousands of labels in the classification degrades the performance of any system [8, 9, 40, 41]. Our system on the other hand is a multilabel classification system that depends on assigning multiple keywords to documents. Our system showed the ability of dealing with the huge label set problem. More importantly, our system can suggest other related keywords to the authors that may not come to the mind.

Additionally, the overfitting problem is another serious challenge in some machine and deep learning models [42–44]. In that case, our system is adaptive in a way you do not have to retrain your system to match new coming data samples. That is, each new incoming data is indexed by the IRS and is ready to be considered in the next query. Thus, unlike machine learning techniques, we do not need to retrain the model. On the other hand, when the number of labels set, or *n*, is increased the probability of discovering a match between the ground truth and the suggested keywords will increase.

There are some drawbacks of the system which are related to selecting the optimal *k,t* or *n*. In our case, this can only be proved by experimentation. That means, we only observe all the parameters, *k*, *t*, or *n*, during experimentation, and then we make the selection of those parameters according to the optimal results.

Finally, our system can be generalized to be used in other domains including the medical one. The only thing that you need to do is to select a sufficient number of documents and use them as a corpus. And let the Elasticsearch with our *k*-NN and *t*-NN methods do the rest. Eventually, the efficiency of the system will depend on the former mentioned parameters.

6. Conclusion

Assigning keywords to research articles is a very important and challenging process. Keywords should properly describe the main concepts that each article may have. Doing this manually is difficult and may cause misidentification of the article's keyword set. Hence, it is a good challenge to address creating a system that will assign multiple keywords automatically. In our study, we designed and developed an automated

keyword assignment system (AKAS) for medical research articles. We used the Medline dataset in the PubMed system as our main medical data source. However, we worked on two different collection-based methods that are accompanied with Elasticsearch information retrieval system for keyword suggestion. First, the proposed keyword suggestion system considers the abstract of the research article as a query for the information retrieval system. The information retrieval system then returns a list of articles based on the ordered similarity score of the given query. Next, we selected an article set from this list using two different methods: k-NN and t-NN. These two methods represent the first k articles and articles whose similarity score is greater than the threshold value of t respectively. To evaluate the proposed system, we conducted a series of experiments using a thousand of randomly selected articles from Medline corpus and compared the system results with the authors' keywords. Our results show that the proposed model offers approximately 55% matching keywords in terms of F-measure. However, we cannot assume that the mismatched keywords are irrelevant. As a result, the researchers will be able to assign appropriate keywords for their articles. Our system can also suggest other external keywords that are not in the authors' original keyword list. One possible future work is to determine the optimal number of keywords n, which is not discussed in this study. For further studies, keywords can be suggested by reordering the ranking result set. A clear challenge of the proposed technique would be to semantically order the keywords. Studies in this direction can also be investigated in the future.

References

[1] Abilhoa WD, De Castro LN. A keyword extraction method from twitter messages represented as graphs. Applied Mathematics and Computation. 2014; 240: 308-325.

[2] Zhang K, Xu H, Tang J, Li J. Keyword extraction using support vector machine. In international conference on web-age information management; Berlin, Heidelberg; 2006; 85-96.

[3] Sarkar K. Automatic keyphrase extraction from medical documents. In International Conference on Pattern Recognition and Machine Intelligence; Berlin, Heidelberg; 2009; 273-278.

[4] Huang Z, Xie Z. A patent keywords extraction method using TextRank model with prior public knowledge. Complex & Intelligent Systems. 2022; 8 (1): 1-2.

[5] Papagiannopoulou E, Tsoumakas G, Papadopoulos A. Keyword Extraction Using Unsupervised Learning on the Document's Adjacency Matrix. InProceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15) 2021; 94-105.

[6] Miah M, Sulaiman J, Sarwar TB, Zamli KZ, Jose R. Study of keyword extraction techniques for electric double-layer capacitor domain using text similarity indexes: An experimental analysis. Complexity. 2021.

[7] Liu Z, Huang W, Zheng Y, Sun M. Automatic keyphrase extraction via topic decomposition. In Proceedings of the 2010 conference on empirical methods in natural language processing; 2010; 366-376.

[8] Ahmed N, Dilmaç F, Alpkocak A. Classification of Biomedical Texts for Cardiovascular Diseases with Deep Neural Network Using a Weighted Feature Representation Method. In Healthcare 2020; 8 (4): 392.

[9] Bi W, Kwok J. Efficient multi-label classification with many labels. In International conference on machine learning. PMLR; 2013; 405-413.

[10] Zhang C. Automatic keyword extraction from documents using conditional random fields. Journal of Computational Information Systems 2008; 4 (3): 1169-1180.

[11] Sebastiani F. Machine learning in automated text categorization. ACM computing surveys (CSUR) 2002; 34 (1): 1-47.

[12] Wang Z, Joshi S, Savel'ev S, Song W, Midya R et al. Fully memristive neural networks for pattern classification with unsupervised learning. Nature Electronics 2018; 1 (2): 137-45.

[13] Van Engelen JE, Hoos HH. A survey on semi-supervised learning. Machine Learning 2020; 109 (2): 373-440.

[14] Matsuo Y, Ishizuka M. Keyword extraction from a document using word co-occurrence statistical information. Transactions of the Japanese Society for Artificial Intelligence 2002; 17 (3):217-23.

[15] Ruch P, Geissbühler A, Gobeill J, Lisacek F, Tbahriti I et al. Using discourse analysis to improve text categorization in MEDLINE. Studies in health technology and informatics 2007; 129 (1):710.

[16] Pay T, Lucci S. Automatic keyword extraction: An ensemble method. In2017 IEEE international conference on big data (big data); 2017. pp. 4816-4818.

[17] Singhal A, Sharma DK. Keyword extraction using Renyi entropy: a statistical and domain independent method. In2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS);IEEE 2021; 1970-1975.

[18] Onan A, Korukoğlu S, Bulut H. Ensemble of keyword extraction methods and classifiers in text classification. Expert Systems with Applications 2016; 57: 232-47.

[19] Hulth A. Improved automatic keyword extraction given more linguistic knowledge. In Proceedings of the 2003 conference on Empirical methods in natural language processing 2003; 216-223.

[20] Rose S, Engel D, Cramer N, Cowley W. Automatic keyword extraction from individual documents. Text mining: applications and theory 2010; 1: 1-20.

[21] Beliga S. Keyword extraction: a review of methods and approaches. University of Rijeka, Department of Informatics, Rijeka. 2014;1 (9).

[22] Gokalp O, Tasci E, Ugur A. A novel wrapper feature selection algorithm based on iterated greedy metaheuristic for sentiment classification. Expert Systems with Applications. 2020; 146: 113176.

[23] Rak R, Kurgan LA, Reformat M. Multilabel associative classification categorization of MEDLINE articles into MeSH keywords. IEEE engineering in medicine and biology magazine. 2007; 26 (2): 47.

[24] HaCohen-Kerner Y, Gross Z, Masa A. Automatic extraction and learning of keyphrases from scientific articles. In International Conference on Intelligent Text Processing and Computational Linguistics. Berlin, Heidelberg 2005; 657-669.

[25] Turney PD. Learning algorithms for keyphrase extraction. Information retrieval 2000; 2 (4): 303-36.

[26] Vega-Oliveros DA, Gomes PS, Milios EE, Berton L. A multi-centrality index for graph-based keyword extraction. Information Processing & Management. 2019; 56 (6): 102063.

[27] Jones KS. Information retrieval and artificial intelligence. Artificial Intelligence. 1999; 114 (1-2): 257-81.

[28] Xiong A, Liu D, Tian H, Liu Z, Yu P et al. News keyword extraction algorithm based on semantic clustering and word graph model. Tsinghua Science and Technology. 2021; 26 (6): 886-93.

[29] Litvak M, Last M. Graph-based keyword extraction for single-document summarization. In Coling 2008: Proceedings of the workshop multi-source multilingual information extraction and summarization 2008. pp. 17-24.

[30] Faraj A, Rashid B, Shareef T. Comparative study of relational and non-relations database performances using Oracle and MongoDB systems. International Journal of Computer Engineering and Technology (IJCET) 2014; 5 (11): 11-22.

[31] Jung MG, Youn SA, Bae J, Choi YL. A study on data input and output performance comparison of mongodb and postgresql in the big data environment. In 2015 8th international conference on database theory and application (DTA). IEEE; 2015. pp. 14-17.

[32] Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. Transactions of the association for computational linguistics 2017; 5: 135-46.

[33] Řehůřek R, Sojka P. Gensim—statistical semantics in python. Retrieved from genism. org. 2011.

[34] Neumann M, King D, Beltagy I, Ammar W. ScispaCy: fast and robust models for biomedical natural language processing. arXiv preprint arXiv:1902.07669. 2019.

[35] Kononenko O, Baysal O, Holmes R, Godfrey MW. Mining modern repositories with elasticsearch. In Proceedings of the 11 th working conference on mining software repositories 2014. pp. 328-331.

[36] Lokhande PS, Aslam F, Hawa N, Munir J, Gulamgaus M. Efficient way of web development using python and flask.International Journal of Advanced Research in Computer Science 2015; 6 (2):54–57.

[37] Vyshnavi VR, Malik A. Efficient Way of Web Development Using Python and Flask. Int. J. Recent Res. Asp 2019;6 (2):16-9.

[38] Tarekegn A, Ricceri F, Costa G, Ferracin E, Giacobini M. Predictive modeling for frailty conditions in elderly people: machine learning approaches. JMIR medical informatics. 2020; 8 (6): e16678.

[39] Tarekegn AN, Giacobini M, Michalak K. A review of methods for imbalanced multi-label classification. Pattern Recognition. 2021; 118: 107965.

[40] Babbar R, Schölkopf B. Data scarcity, robustness and extreme multi-label classification. Machine Learning. 2019; 108(8): 1329-51.

[41] Hu M, Han H, Shan S, Chen X. Multi-label learning from noisy labels with non-linear feature transformation. InAsian Conference on Computer Vision. Springer, Cham; 2018. pp. 404-419

[42] Peng YL, Lee WP. Data selection to avoid overfitting for foreign exchange intraday trading with machine learning. Applied Soft Computing. 2021; 108: 107461.

[43] Bejani MM, Ghatee M. A systematic review on overfitting control in shallow and deep neural networks. Artificial Intelligence Review. 2021; 54(8): 6391-438.

[44] Chen CC, Watabe M, Shiba K, Sogabe M, Sakamoto K, Sogabe T. On the expressibility and overfitting of quantum circuit learning. ACM Transactions on Quantum Computing. 2021 Jul 9;2(2):1-24.