COVID-19 Literature Mining and Retrieval Using Text Mining Approaches

SANKU, Satya Uday, PAVANI, Satti Thanuja, TANGIRALA, Jaya Lakshmi <http://orcid.org/0000-0003-0183-4093> and CHIVUKULA, Rohit

COVID-19 Literature Mining and Retrieval using Text Mining Approaches

Satya Uday Sanku[1], Satti Thanuja Pavani[1], T. Jaya Lakshmi[1*], Rohit Chivukula[2]

[1*]Department of Computer Science and Engineering, SRM University, Guntur, 522502, Andhra Pradesh, India.
[2]School of Computing, University of Huddersfiled, Huddersfield, HD1 3DH, , United Kingdom.

*Corresponding author(s). E-mail(s): jaya.phd.hcu@gmail.com;
Contributing authors: satya_sanku@srmap.edu.in;
satti_thanuja@srmap.edu.in; rohit.chivukula.kingdom@gmail.com;

Abstract

In light of the recent COVID-19 epidemic, users are facing growing difficulties in navigating the vast expanse of internet content in order to locate relevant information. In this study, we have developed an information extraction mechanism to address users' inquiries pertaining to COVID-19, catering to a range of depths in response. In order to accomplish this objective, the CORD-19 dataset, which has been made available by the Allen Institute for AI, is utilized. This dataset comprises 200,000 scholarly articles that pertain to research papers on the topic of coronavirus. These articles have been sourced from many reputable platforms such as PubMed's PMC, WHO, bioRxiv, and medRxiv pre-prints. In addition to the aforementioned document corpus, a supplementary list of topics has been furnished, encompassing inquiries pertaining to the infection. Each topic consists of three levels of representations, namely query, question, and story. Inquiry can take on different forms, with query representing a fundamental form, question serving as a more intermediate form, and narrative embodying a more detailed and elaborate type of inquiry. The proposed model uses various word embedding techniques such as frequency based (Bag-of-words), semantic based (Word2Vec), a hybrid method which combine frequency with semantic (TF-IDF weighted Word2Vec) as well as sequence cum semantic based (BERT) to fabricate vectors for the documents in the corpus, query, question, narrative, and combinations of them. Once vectors have been created, cosine similarity is employed to identify similarities between document vectors and topic vectors. As compared

to frequency and semantic models, BERT demonstrates a higher degree of relevance in retrieving documents. with 90% accuracy. The proposed hybrid model which is the TF-IDF weighted Word2Vec, achieves an accuracy rate of 87%. This is comparable to the average performance of the Bert-Base model demonstrating computational efficiency.

Keywords: Information extraction, word-embedding models, Bag-of-Words, Word2Vec, BERT-Base model, TF-IDF, Weighted Word2Vec.

1 Introduction

As the pandemic has made its way to an extreme level, researchers worldwide are working to know the behavior of the virus and propose ways to impede the spread of the virus, therefore lakhs of research papers are generated. This massive generation of data makes it tough to find the relevant topic of interest for the researchers [1]. A huge number of research articles published related to Covid-19 are made available in the CORD-19 dataset by Allen Institute for AI [2]. However, all of this data is presented in a human-readable format, making it challenging for computer algorithms to process. Therefore, it is necessary to translate this text data into computer-friendly form. Natural Language Processing (NLP) is a subfield of artificial intelligence that deals with computers and human language interactions, specifically how to teach a computer to process and analyse huge amounts of natural language input in a human-readable form [3]. This enables the computer to comprehend the content and context of papers.

The primary objective of this study is to provide a compilation of pertinent document titles for user interested topics expressed in the form of queries. It is necessary to convert the titles and topics conveyed in the English language into a format that is recognizable by computers. Natural Language Processing (NLP) approaches play a significant role in facilitating this transformation. Once they are converted into computer friendly format, similarity can be compared between topics and titles to return the appropriate research articles containing the relevant titles. A plethora of natural language processing (NLP) models may be found in the existing literature. This study utilizes four models. A comprehensive analysis is provided regarding the applicability and limitations of each model.

1.1 Problem Statement

Given a large corpus of research documents related to COVID-19, each containing title and abstract and a list of topics at various granularity referred as query, question and narrative, the goal is to extract relevant research documents related to topic from the corpus.

The problem is challenging because there are multiple levels of granularity for both the document and the topic being searched for. This study proposes models for predicting the documents corresponding to titles for topics expressed at three levels:

query, question and narrative. This enables researchers to acquire more accurate titles and abstracts for their queries.

The rest of the paper is organised as follows: Section 2 provides the details of dataset used in this work. Section 3 presents the related literature. In Section 4, we discuss the steps of the proposed method, including the preprocessing, the word embedding models employed, and the specifics of the performance evaluation. Section 5 concludes the work.

2 Dataset Description

An information retrieval challenge with the name Trec-Covid has been organised by the Allen Institute for Artificial Intelligence, National Institute of Standards and Technology, National Library of Medicine, Oregon Health and Science University, and the University of Texas Health Science Center at Houston on the popular coding challenge platform Kaggle.com [2]. Trec-Covid dispenses us with three data files named as metadata.csv (CORD-19 dataset), topics-rnd3.csv and qrels.csv.

Metadata.csv contains a unique identifier of each document called as *cord_uid*; along with the corresponding titles and abstracts. The abstract gives an extensive description of the content of the document. It also contains other features such as the *pubmed_id* of the authors and the *publish_time*. The metadata file consists of 1.3 lakh titles and abstracts.

The file topics-rnd3.csv contains topic-id, query, question, and narrative. The query is a short description of a topic, and the question is a slightly extensive description of a query, where as the narrative describes the question more in detail. A total of 40 topics have been made available by the challenge.

The file qrels.csv contains the relevant judgments for documents that have been evaluated in previous rounds.

The CORD-19 dataset consists of 103,000 documents in total. The dataset description is given in Table.1 and sample records in these three files are given in Tables 2,3 and 4.

3 Related Literature

Machine learning is emerging as one of the hot topics in the industry as well as in academic institutions to smoothen the analytical tasks with huge amount of data being available daily. This data is mostly in the form of unstructured text documents [4]. There are algorithms in abundance for performing various analytical activities in order to gain actionable insights from this big data with the help of natural language processing (NLP) paradigm [4],[5]. Data processing takes a long time, as demonstrated by the work of Long Ma et al., not only due to the sheer volume of information being processed, but also due to the variety and complexity of the data itself [6]. Data mining and machine learning approaches are being used to tackle the challenge of massive amounts of data. These algorithms work effectively by choosing useful features. The methods for structuring text data that have been described in the literature will be explored in the next section.

Table 1 CORD-19 dataset description

File	Description	#Records	#Features	Features
metadata.csv	Information about research articles published on Covid-19	103,000	19	cord_id title abstract pubmed_id publish_time ...
topics-rnd3.csv	Queries to retrieve from documents at different levels of granularity.	40	4	topic_id query question narrative
qrels.csv	Relevant judgments for documents that have been evaluated in previous rounds. Three possible values of judgements: 0(non-relevant); 1(partially relevant) and 2(relevant).	20,728	4	topic_id iteration cord_id judgement

Table 2 Sample documents in CORD-19 Dataset (metadata.csv)

Document#	**cord_uid**	**Title**	**Abstract**
117	rqkgrd2k	ebola virus come going	long text
118	q1q5801p	essentials pulmonology	long text
119	p6c57zzm	general mechanisms antiviral resistance	long text

Table 3 Sample documents in qrls.csv of CORD-19 Dataset

topic-id	**iteration**	**cord-id**	**judgement**
1	0.5	010vptx3	2
1	1.0	02f0opkr	1
1	1.0	04ftw7k9	0
1	1.0	05qglt1f	0

3.1 Structured representation of text data

We use four different word embedding Natural Language Processing models in this work: Frequency based; Semantic based; Frequency+Semantic based and Sequence+Semantic based.

Bag of Words (BoW) is most widely used frequency based technique for converting unstructured text to structured [7]. BoW constructs fixed size numerical vector for each text document in the document corpus [8]. BoW initially builds a list of words from all the documents in the corpus. Then, for each document, a numerical vector is constructed as the vector containing the count of occurrences of a word in that document.

Term Frequency and Inverse Document Frequency, or TF-IDF, is another popular frequency bases NLP technique for generating vector representations of text data

Table 4 Sample record in CORD-19 Dataset

Feature	value
topic-id	1
query	coronavirus origin
question	what is the origin of COVID-19
narrative	seeking range of information about the SARS-Co... (long text)

[9]. This method first computes Term Frequency(TF) and Inverse Document Frequency(IDF). By multiplying the TF and IDF values, TF-IDF generates a document vector. The frequency of a certain term in the document is examined using term frequency. Frequency can be defined in a variety of ways. The inverse document frequency measures the frequency with which a term appears in the corpus as a whole. It is frequently calculated as the ratio of total documents to papers containing the phrase t.

A lot of pre-processing is involved in this computation depending on the domain. The details of pre-processing performed in this work is present in Section 4.

A dense word vector form is constructed using word embedding. Words that are similar in meaning and vocabulary are close to each other in the embedding space[10]. Word2Vec belongs to the class of Semantic based word embedding algorithms [11],[12]. Word2Vec is a Google-proposed and funded algorithm [6].

BERT uses the sequence of words in the text with semantics and is a trained Transformer Encoder stack [13]. BERT receives a series of words as input, which continue to flow up the stack [13]. Each encoder performs self-attention, transfers the results through a feed-forward network, and then passes the information on to the next encoder. After reaching the last encoder the Bert base model releases the embedding for each word present in the sentence based on the context. The output will be a high dimensional word embedding for each word. So for every sentence the number of word tensors depends upon the number of words present in that sentence.

Once the document vectors are constructed using the models described, the similarity between these numerical vectors can be computed using distance/similarity measures. The following section describes the popular similarity measures.

3.2 Similarity measures

Similarity is often assessed using a distance metric such as Euclidean distance or Cosine similarity. Euclidean distance is the distance between two points in the Euclidean space and can be conveniently calculated between multi-dimensional points. This is the most common ly used metric in text clustering [14]. Euclidean distance computes square root of summation of squared distance between the corresponding vector features. Euclidean distance between a pair of vectors (X,Y) where X=(x_1,x_2,... x_n) and Y=(y_1,y_2,... y_n) can be computed using the equation (1).

$$Euclidean\_dist(X,Y) = \sqrt{\sum_{i-1}^{n}(x_i - y_i)^2} \tag{1}$$

Cosine similarity is another popular measure. Cosine similarity captures the correlation between the document vectors. This is expressed as the cosine of the angle

formed by two vectors, or cosine similarity [15, 16]. Cosine similarity measures the angular distance between two vectors [17] which is shown in Fig.1.

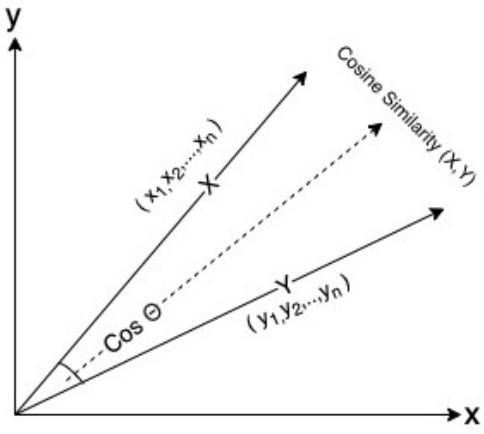

Fig. 1 Cosine Similarity

Cosine similarity between a pair of vectors (X,Y) can be computed between two vectors X=(x_1,x_2,... x_n) and Y=(y_1,y_2,... y_n) using the equation (2).

$$\begin{aligned} cos(x,y) &= \frac{X.Y}{||X||_2||Y||_2} \\ X.Y &= x_1.x_2 \dots x_n + y_1.y_2 \dots y_n \\ ||X||_2 &= x_1^2 + x_2^2 \dots x_n^2 \\ ||Y||_2 &= y_1^2 + y_2^2 \dots y_n^2 \end{aligned} \tag{2}$$

Cosine similarity values range between 0 and 1. If the cosine similarity is zero, then the two documents being compared are dissimilar. The two documents are closely similar if cosine similarity is 1.

As cosine similarity is more popular in the area of text mining, we use cosine similarity in this work.

A few more interesting works in the literature can be found in [18], [19], [20], [21], [22], [23], [24] and [25].

The proposed approach to construct document vectors is given in section 4 and each step in detail is explained in further sections.

4 Proposed Approach

The primary objective of this work is to generate a list of document titles that are most closely related to the given topics(user queries). The proposed approach considers all the titles and topics and preprocess them. The preprocessed titles and topics are then transformed into vectors. After pre-processing, the titles and topics are converted into numerical vectors of same size using different word embedding models. Then the similarity of each titles and topics is computed. At last, the top 'k' titles relevant to the topics are returned. This proposed approach is depicted in Fig. 2.

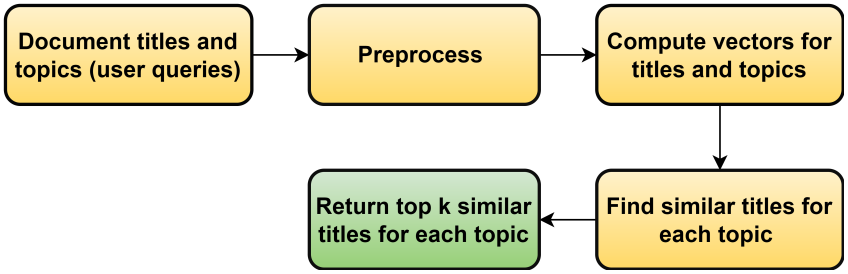

Fig. 2 Framework of Proposed Approach

As already mentioned earlier, topics(interested to the user) are stated at various granularities. Query is the most basic form of inquiry; questions are intermediate; and narratives are the most in-depth expressions of topic. When we talk about "topic", we're referring to all three of these formats. When we refer to a topic, we mean all three levels at once, however each is processed/computed separately.

We use four kinds of word-embedding approaches to convert the text to vectors. Word embedding is a language modelling technique to represent the words or phrases as vectors of real numbers. Once the vectors are computed, the similarity between each topic vector is computed against title vector. The top k most similar titles are then returned as an answer to the topic. As different text mining methods are used in this task, performance of each technique applied is evaluated using standard metrics.

Steps in the approach are detailed as

- Preprocessing
- Apply NLP model
- Evaluate performance of the model

This detailed proposed approach is given in Fig. 3. Each step is explained in detail in the following sections.

4.1 Preprocessing

Titles in CORD-19 dataset need to be preprocessed for making it suitable for retrieval process. There are a few null titles observed in the CORD-19 dataset. After removing these null titles, 1,28,459 titles are left in the dataset. from the non-empty titles, special characters and stop-words are eliminated. There are 6292 duplicate titles and 18064 repeats after this point in the analysis. This is happening because existence of some special characters generate two separate words that are occasionally the same. One such situation is that in the original corpus, words inside a bracket asn the same word without being in brackets are regarded as distinct. For example, "[Algemene leading]" and "Algemene leading" are treated as two different titles since the first title contains special characters at the beginning and ending. The same kind of preprocessing is applied for queries, questions and narratives also.

There are a few Chinese titles in the given data set, but our processing is all on English language titles, hence Chinese titles will not fit into our processing. Therefore, we have decided to eliminate Chinese titles from the corpus. Text elimination is

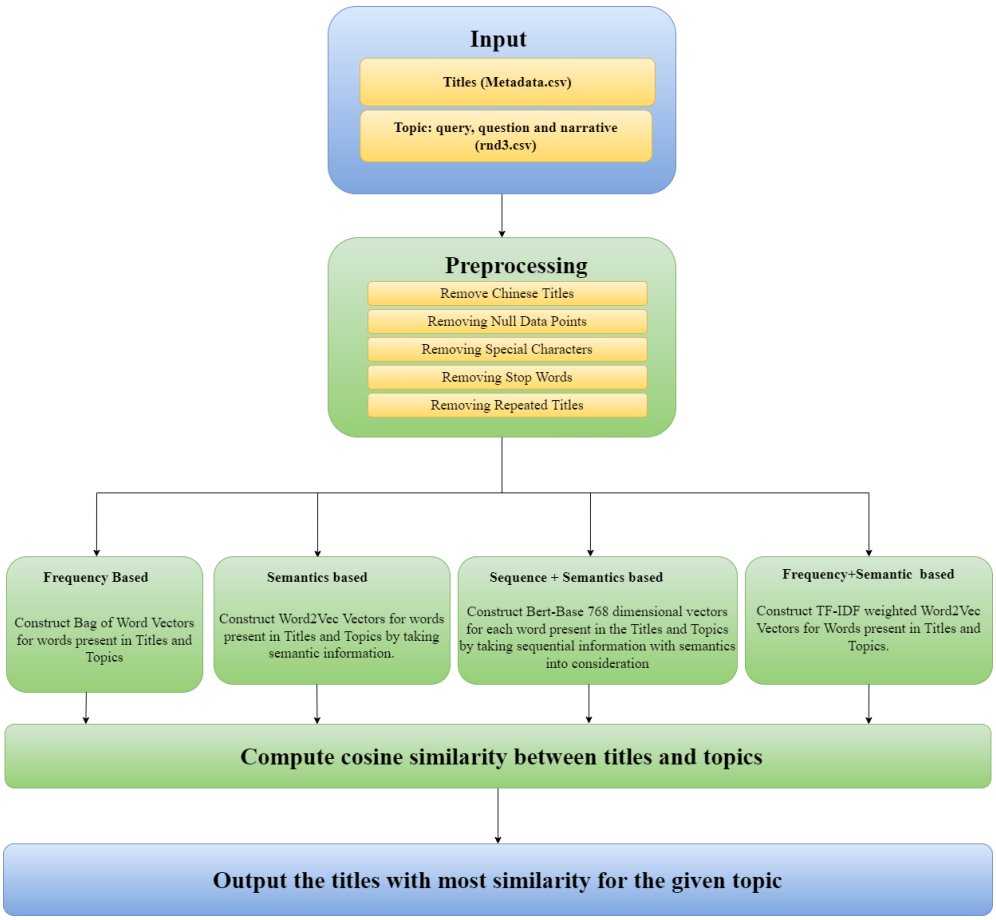

Fig. 3 Proposed Approach

commonly done using text comparison. But it is difficult to use the same process for non-English titles. Therefore utilizing ASCII encoding is useful in such scenarios. An example of unreadable text in Chinese language taken from our experimentation is given in Fig.4

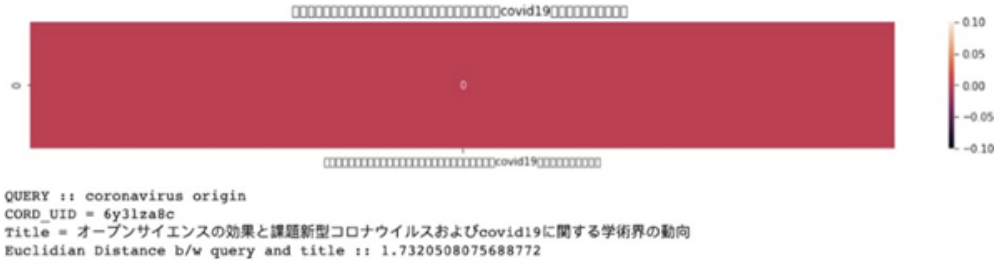

Fig. 4 Chinese Titles

Once the tasks of removing null and duplicate and Chinese titles, porter stemmer [26] is used for converting all the words in the titles into a root form. Then we apply Lemmatization to further process.

Lemmatization usually refers to use a vocabulary and morphological analysis of words, typically aiming to remove inflectional endings only and return the base or dictionary form of the word [26].

Lemmatization exhibits superior accuracy in comparison to stemming. Lemmatization is the ideal method for conducting context analysis, but stemming is commonly advised in cases when the context holds less significance. The process of stemming involves the removal of word ends without taking into account the language context, resulting in improved computer efficiency. Lemmatization involves the analysis of word forms in order to identify their base or dictionary form, a procedure that requires additional computational resources. The utilization of lemmatization is favored in this study due to the heightened significance of accuracy in queries pertaining to the medical domain.

We use Lemmatization for converting all the words in the titles as well as queries, narratives, and questions.

The title and topic are to be converted into numerical vectors of uniform size. The conversion of a document containing words into numerical vector has the following steps:

- Compute a $word\_list$ containing all words in all titles and topics. Let the length of the $word\_list$ be n.
- For each title/topic,
  - Create a vector of size n where indices are the words.
  - Compute a numeric score for $vector[i]$ based on the existence of the $word\_list[i]$ in the title/topic. A simple strategy can be marking 1 in $vector[i]$ if $word\_list[i]$ exists in that title/topic, mark 0 otherwise.

Different word-embedding models compute the numeric vector in different ways. The following four different word embedding Natural Language Processing models are used in this work to generate the vectors.

- Frequency based
- Semantic based

- Frequency+Semantic based
- Sequence+Semantic based

The in depth working of these four models is given in next four sections.

4.2 Frequency based word embedding model

Frequency-based embedding models take into account the frequency with which certain words appear in the text. Bag-of-Words and TF-IDF are two popular methods in this category. We discuss BoW in this section and TF-IDF in Section. 4.4.

Bag-of-Words

The vectors of topic and title using Bag-of-Words method are computed as follows. First of all, the preprocessed topics and titles are amalgamated into a single list of words. Taking these words as features, the count of these words in every topic and title is computed. Note that, every title and topic's Bag of words vector has same dimension, facilitating the computation of similarity score. At first, we compute BoW vector considering words in title and query are computed. Then we expand BoW vectors to include words in question. St last, we extend the model also to include words in narrative. Thus, we use different combinations of query, question and narrative information to compute BoW vectors. The method that includes all the three is summarised in Algorithm. 1. However, we analyse the performance with different combinations.

We compute cosine similarity of titles with different combinations of query, question and narratives. The results are discussed in Section 5.

4.3 Semantic based word embedding model

Word2vec(W2V) is another word embedding technique that takes semantics of the word into consideration. W2V algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence [27]. Large-scale word embedding learning is possible with the W2V modelling framework, which consists of a variety of model topologies and optimization techniques. The two most widely used strategies are as follows:

- Continuous Bag of words model [28]: The CBOW model attempts to comprehend the context of the words and uses it as input. It then attempts to forecast contextually relevant terms. If users don't concern about the order in which words appear in the context, then this architecture is simply a bag-of-words model.
- Skip-gram model [29]: Anticipates words before and after current word in the same phrase based on a given range of words in the same range.

In this work, we use CBoW variation of W2V model to fabricate the W2V's for every word in the given corpus (titles, queries, questions, narratives). Using this trained W2V model, for each word in a title, 100 dimensional W2V's are fabricated. As in all above methods, we use different combinations of topics. Then cosine similarity is computed between titles and topics.

Algorithm 1 Computation of BoW for titles and topics

Input:

t: A list of n titles; m: Number of Topics; qr: A list of m queries; qs: A list of m questions and nr: A list of m narratives.

Output:

bow_title, *bow_query*, *bow_question* and *bow_narrative*: A list of numerical vectors for t, qr, qs and nr.

word_list=empty list
for each *title* in t **do**
  for each *word* in *title* **do**
    if *word* not in *word_list* **then**
      Add *word* to *word_list*
    end if
  end for
end for
for each *query* in qr, *question* in qs and *narrative* in nr **do**
  for each *word* in *query* **do**
    if *word* not in *word_list* **then**
      Add *word* to *word_list*
    end if
  end for
  for each *word* in *question* **do**
    if *word* not in *word_list* **then**
      Add *word* to *word_list*
    end if
  end for
  for each *word* in *narrative* **do**
    if *word* not in *word_list* **then**
      Add *word* to *word_list*
    end if
  end for
end for
num=$|$*word_list*$|$
for each i in $1 \ldots num$ **do**
  bow_title[i]=count of *word_list*[i] in *title*
  bow_query[i]=count of *word_list*[i] in *query*
  bow_question[i]=count of *word_list*[i] in *question*
  bow_narrative[i]=count of *word_list*[i] in *narrative*
end for

Return *bow_title*, *bow_query*, *bow_question* and *bow_narrative*

4.4 Frequency + Semantic based

We use a fusion of TF-IDF, which is frequency based word embedding and W2V, which is semantic based model as our next model in this work inspired by the work of [30]. The procedure of computing TF-IDF vectors is given in next section.

4.4.1 TF-IDF

TF-IDF stands for Term Frequency(TF) and Inverse Document Frequency(IDF) which is an NLP technique used to create vector representation for the given textual data. Firstly we calculate the TF values that gives us the information about how frequent the word W_j in the given text T_i. Then IDF values are computed which gives the information about how less frequent or how rare a word occurs in Document Corpus. For Instance, if the word repetition is high in the given text, Term Frequency(TF) value increases. while, in case of Inverse Document frequency, IDF value is high if W_j occurrence is rare in given document corpus. Both TF and IDF values are multiplied to get TF-IDF values. TF-IDF gives larger values for less frequent words in the document corpus. The procedure we use to compute TF-IDF vectors of titles and topics is given in Algorithm. 3.

After computing TF-IDF values for each word they are stored in the vector by considering words as feature names in the given review text. We combine TF-IDF with W2V models as follows:

1. Initially, for each title and topic in the Data Corpus, two distinct vectors are generated: the TF-IDF vector and the Word2Vec vector.
2. The TF-IDF values for every word within each document are computed as outlined in Section 4.4.1. This step assigns a weight to each word based on its importance within the context of the document.
3. Simultaneously, W2V vectors are calculated for every individual word within the text, following the procedure elaborated in Section 4.3. These W2V vectors capture the semantic meaning of each word.
4. The next step is where the normalization process comes into play. To obtain TF-IDF weighted Word2Vec vectors, the corresponding elements in the TF-IDF vector and the Word2Vec vector for each word are multiplied together. This operation combines the semantic meaning of each word (from W2V) with its importance in the document (from TF-IDF), effectively producing a TF-IDF weighted W2V vector for each word within the title or topic.
5. However, to achieve uniformity and ensure that the length or magnitude of these TF-IDF weighted W2V vectors does not vary significantly based on the length of the title or topic, normalization is carried out. This is accomplished by dividing the total TF-IDF weighted Word2Vec vector (sum of all word-level TF-IDF weighted W2V vectors) by the total number of words in the text (title or topic). The result is a normalized TF-IDF weighted Word2Vec vector that represents the entire text, capturing both the semantic information of the words and their importance within the context of the document.

Cosine similarity is used to obtain the similarities between topics and titles.

Algorithm 2 Computation of $TF - IDF$ vectors for titles and topics

Input:

t: A list of n titles

m: Number of Topics

qr: A list of m queries

qs: A list of m questions and

nr: A list of m narratives.

Output:

tf_idf_title, tf_idf_query, tf_idf_ques and tf_idf_nar: A list of numerical vectors for t, qr, qs and nr.

D=T U qr U qs U nr

$word\_list$: A list containing all the words in T, qr, qs and nr

IDF=*empty list*

for each $word$ in $word\_list$ **do**

$IDF[word] = log(\frac{|D|}{No\ of\ texts\ containing\ word\ in\ D})$

end for

num=$|word\_list|$

for each $word$ in $word\_list$ **do**

for each t in T, q_1 in qr, q_2 in qs and q_3 in nr **do**

$TF[word, t] = \frac{Number\ of\ times\ word\ occurs\ in\ t}{Number\ of\ words\ in\ t}$

$tf\_idf\_title[word]$=$TF[word, t] * IDF[word]$

$TF[word, q1] = \frac{Number\ of\ times\ word\ occurs\ in\ q_1}{Number\ of\ words\ inq_1}$

$tf\_idf\_query[word]$=$TF[word, q1] * IDF[word]$

$TF[word, q2] = \frac{Num\ of\ times\ word\ occurs\ in\ q_2}{Number\ words\ in\ q_2}$

$tf\_idf\_ques[word]$=$TF[word, q2] * IDF[word]$

$TF[word, q3] = \frac{Number\ of\ times\ word\ occurs\ in\ q_3}{Number\ of\ words\ in\ q_3}$

$tf\_idf\_nar[word]$=$TF[word, q3] * IDF[word]$

end for

end for

Return tf_idf_title, tf_idf_query, $tf\_idf\_question$ and $tf\_idf\_narrative$

Algorithm 3 Computation of $TF-IDF$ weighted $W2V$ for titles and topics

Input:
tf_idf: TF-IDF vectors of titles and topics
$w2v$: W2V vectors of titles ad topics

Output:
tf_idf_w2v: A list of numerical vectors for titles and topics

Obtain *word_list*: A list containing all the words in *titles* and *topics*
num=$|word\_list|$

for each *word* in *word_list* **do**
  for each t in *title* **do**
    nw=Number of words in t
    $tf\_idf\_w2v[word,t]=\frac{tf\_idf[word,t]*w2v[[word,t]}{nw}$
  end for
  for each t in *topic* **do**
    nw=Number of words in t
    $tf\_idf\_w2v[word,t]=\frac{tf\_idf[word,t]*w2v[[word,t]}{nw}$
  end for
end for
Return tf_idf_w2v

4.5 Sequence+Semantic based word embedding

This class of models consider both context and semantics of the word in the text. BERT [13] is an excellent example of this class of models.

BERT(Bidirectional Encoder Representations from Transformers) is basically a trained Transformer Encoder stack. We use 12 encoder layers in the BERT model in this work. These have larger feedforward-networks with 768 hidden units and 12 attention heads. BERT receives a series of words as input, which continue to flow up the stack [13]. Each encoder performs self-attention, transfers the results through a feed-forward network, and then passes the information on to the next encoder. After reaching the last encoder the Bert base model will release the embeddings for each word present in the sentence based on the context. The output will be 768 dimensional word embedding for each word. So for every sentence the number of word tensors depends upon the number of words present in that sentence. But In-order to compare two sentences, the embedding for the entire sentence is required. For computing this, all the word tensors of each sentence are averaged to get 768 dimensional sentence embedding, which is unique in terms of dimension for every sentence in the dataset.

Once the vectors are computed using Average BERT model descibed above, we use cosine similarity to find the closest titles to given queries, questions and narratives. As in the other mothods, we use combinations as well as individual techniques.

4.6 Performance Evaluation Measures

Once the numerical vectors are computed for titles nad topics and relevant documents are obtained for each topic, the relevancy of the titles extracted are evaluated using two metrics in this work: Accuracy and confusion matrix.

Accuracy is defined as

$$Accuracy = \frac{\#Correctly\ classified\ instances}{\#Instances\ in\ test\ dataset} \tag{3}$$

Confusion matrix computes True positive rate, False positive rate, True negative rate and False negative rate. These rates aid in determining where the model is sensibly perplexing. The positive and negative rates provided in the confusion matrix will aid in the computation of relevant metrics such as accuracy, precision, recall, and F1-Score, notably.

Results of this experimentation are discussed in next section.

5 Results

In the competition of Trec-Covid, they provide the relevant judgments for some of the combinations of topics and titles. The given judgments are containing three labels:0,1 and 2 where 0 corresponds to non-relevant, 1 associates to partially-relevant and 2 is used to denote pure-relevant judgement. In this work, we have converted this multi-class problem to binary-class problem by mapping non-relevant judgement to 0 and the other two to 1. We extract top 20 similar titles for every query based on the cosine similarity in bag-of-words, Word2vec, Average Bert-Base model, and Tf-Idf weighted W2V compared with the relevant judgements.

Heatmaps are used to visually represent the common words present in the title and query pairs. Frequency of every word present in the title or query obtained using BoW is shown in the heatmap given in Fig.5.

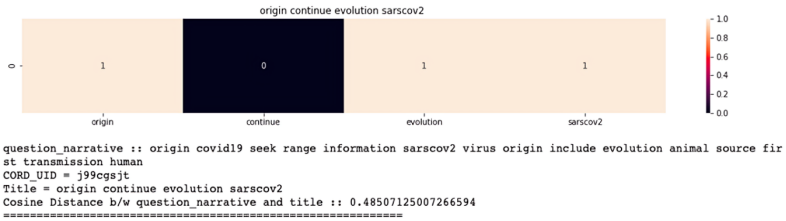

Fig. 5 HeatMap of Bag-of-Words

Using Word2vec, the cosine similarity between each word present in title is compared with every phrase present in respective query and is visually shown in Fig.6. The confusion matrices of both models are given in Fig.7 and Fig.8.

For the bag-of-words model, the accuracy using cosine similarity at the threshold of 0.4 for the text with and without lemmatization are presented in Table. 5.

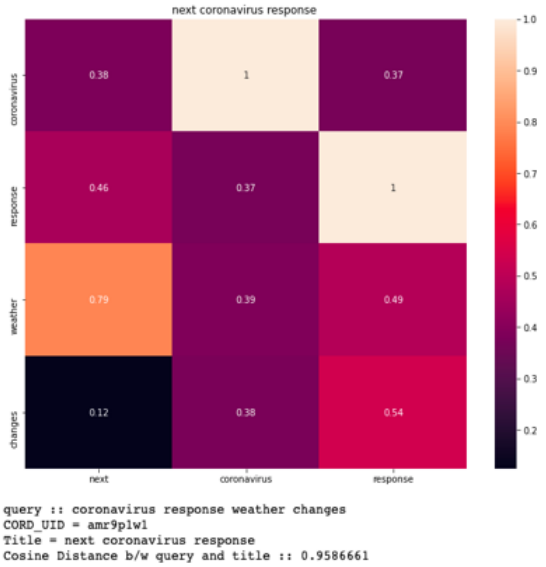

Fig. 6 HeatMap of Word2Vec

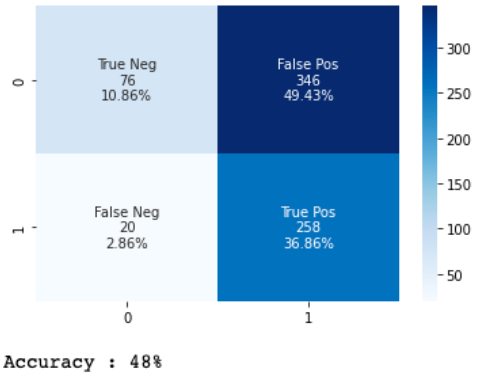

Fig. 7 Confusion Matrix of Lemmatized Bag-of-Words model

Lemmatization did not enhance the model performance in case of Word2Vec as seen in Table. 6. The lemmatized text helped only in frequency based model but not in semantic and sequence based model(Word2Vec). Accuracy of Word2vec for threshold of 0.45 is shown in Table. 6.

Using the Word2vec model, the accuracy ranges from 74% to 78%. Achieving the highest levels of accuracy for both Narrative and Query is particularly rewarding. Confusion Matrix of Word2Vec for the narrative with the title is presented in Fig. 8. When lemmatization is done to the text before the Word2Vec model receives it, unfortunately, the model's accuracy drops. Clearly, Lemmatization works better for frequency-based models like Word2Vec than it does for models that are not frequency-based.

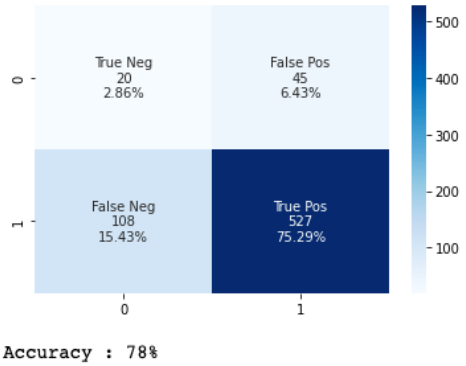

Fig. 8 Confusion Matrix of Word2Vec

Table 5 Accuracy of Bag of Words model for different levels of topic with title at threshold 0.4

TOPICS WITH TITLE	non-LEMMATIZED	LEMMATIZED
Query	46%	48%
Question	24%	29%
Narrative	12%	15%
Query + Question	36%	41%
Query + Narrative	19%	22%
Question + Narrative	16%	20%
Query + Question + Narrative	23%	28%

Table 6 Accuracy of Word2Vec model for different levels of topic with title

TOPIC	non-Lemmatized	Lemmatized
Query	78%	70%
Question	74%	70%
Narrative	78%	70%
Query + Question	77%	71%
Query + Narrative	74%	70%
Question + Narrative	77%	67%
Query + Question + Narrative	74%	72%

As demonstrated in Table. 7, the Average Bert-Base Model predicted meaningful answers to queries, questions and narratives, as well as had a better level of accuracy than the other models. The cosine similarity scores of question number 36 and query number 36 in topics dataset are given in Fig. 9.

TF-IDF weighted Word2Vec, however, needed substantially less processing effort to generate vectors from textual data and delivered results close to Average Bert-Base. While the Bag of Words and Word2Vec models don't keep track of word sequence information when creating vectors for textual input, the Bert-Base model does. In terms of vector fabrication, TF-IDF weighted Word2Vec model added more information to Word2Vec vectors. The TF-IDF weighted Word2Vec model outperformed the Bag of

Table 7 Accuracy of Average Bert-Base model

TOPICS	ACCURACY of AVG-BERT-BASE
Query	73%
Question	90%
Narrative	87%
Query + Question	89%
Query + Narrative	86%
Question + Narrative	88%
Query + Question + Narrative	88%

topic-id	cord-id	query	title	Cosine_Similarity
35	uxuvt0cd	coronavirus public datasets	epidemiology genome clinical features pandemic...	0.901961
35	bhhhslsr	coronavirus public datasets	strong evolutionary convergence receptorbindin...	0.821689
35	y4zpl1ji	coronavirus public datasets	first 12 patients coronavirus disease 2019 cov...	0.817920
35	l63z3btv	coronavirus public datasets	receptor recognition novel coronavirus wuhan a...	0.815415
35	z0bkpmpk	coronavirus public datasets	emergence novel coronavirus causing respirator...	0.814370
35	24et9foe	coronavirus public datasets	genetic diversity evolution sarscov2	0.813902
35	6tgl1cur	coronavirus public datasets	outbreak coronavirus disease 2019 covid19an em...	0.809954

topic-id	cord-id	question	title	Cosine_Similarity
35	z0bkpmpk	new public datasets available related covid19	emergence novel coronavirus causing respirator...	0.859471
35	uxuvt0cd	new public datasets available related covid19	epidemiology genome clinical features pandemic...	0.858733
35	op4z052p	new public datasets available related covid19	italian outbreak covid19 conditions contributo...	0.858124
35	4n6v5kfv	new public datasets available related covid19	emerging novel coronavirus 2019ncovcurrent sce...	0.857065
35	5o12mbut	new public datasets available related covid19	phylogenetic analysis structural modeling sars...	0.857063
35	6tgl1cur	new public datasets available related covid19	outbreak coronavirus disease 2019 covid19an em...	0.853317
35	k21bq0qc	new public datasets available related covid19	sars mers thrusting coronaviruses spotlight	0.848162

Fig. 9 Results of Average Bert-Base Model

words and W2V in terms of accuracy while requiring less processing time. When posed questions were compared to the titles in the document corpus, this model achieved the maximum accuracy, which resulted in a superior recommendation of titles for the posed inquiry is depicted in Table. 8.

Table.9 contains the results of all of the NLP models that were utilised in this study. The medium description of the topic(question) extracted relevant document titles than query and narrative in most of the models. Clearly, BERT is the best model for extracting topic-relevant titles. BERT is however computationally intensive. At varying degrees of topic description, the hybrid model that employs both TF-IDF and W2V information and is computationally more efficient than BERT yields nearly identical results. Fig.9 shows histogram of results for better visualization.

Note that, only top k similar documents are returned by the algorithm. If none of the titles are similar, no output is provided.

Table 8 Accuracy of TF-IDF Weighted W2V Model

TOPICS	TF-IDF Weighted W2V ACCURACY
Query	79%
Question	87%
Narrative	70%
Query + Question	85%
Query + Narrative	84%
Question + Narrative	84%
Query + Question + Narrative	85%

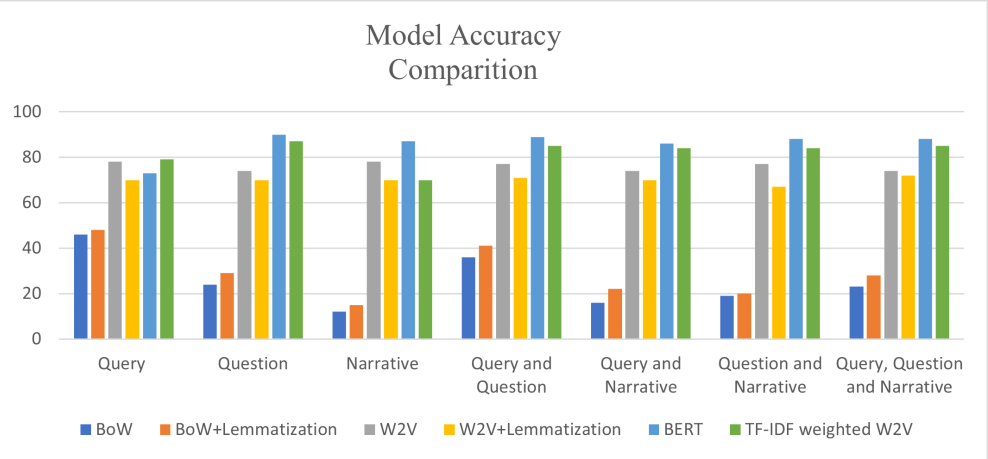

Fig. 10 Results of all word embedding models

6 Conclusion and future work

In this work, we have evaluated various word-embedding methods on CORD-19 dataset relreased by Allien Institute of AI related to COVID-19. Semantic based approach Word2vec comprehends the semantics of words and outperformed the frequency-based technique, Bag of Words. The meaning of the words in a sentence is not sufficient, but the order in which those words appear also plays a crucial influence in determining the accuracy of the results. Fabricating vectors using BERT-base model, that includes both semantics and sequence information leads to better accurate predictions more than the frequency based and semantic based models. However, the computational time of BERT is more than the other models. We employ a hybrid model which is a combination of TF-IDF and Word2Vec. This hybrid model gives the performance near to Average BERT-base model and also computationally efficient. Our future goal is to construct new hybrid models to achieve better and more accurate predictions.

Competing Interests

The authors do not have any Competing Interests.

Table 9 Results of all word embedding models

TOPICS	BoW	BoW with Lemmatization	W2V	W2V with Lemmatization	Bert	Tf-idf weighted W2V
Query	46%	48%	78%	70%	73%	79%
Question	24%	29%	74%	70%	90%	87%
Narrative	12%	15%	78%	70%	87%	70%
Query + Question	36%	41%	77%	71%	89%	85%
Query + Narrative	16%	22%	74%	70%	86%	84%
Question + Narrative	19%	20%	77%	67%	88%	84%
Query + Question + Narrative	23%	28%	74%	72%	88%	85%

Funding Information

No external funding is received for this work.

Author contribution

The main research idea and implementation is of Satya Uday Sanku and Satti Thanuja Pavani equally; Documentation is done by Rohit Chivukula; results are interpretted and analysed by T. Jaya Lakshmi.

Data Availability Statement

Data used in this work is publicly available at kaggle website: https://www.kaggle.com/competitions/trec-covid-information-retrieval

Research Involving Human and/or Animals

Not applicable.

Informed Consent

All authors provided consent for this publication.

References

[1] Heaton, C. T. & Mitra, P. Repurposing trec-covid annotations to answer the key questions of cord-19. *arXiv preprint arXiv:2008.12353* (2020).

[2] website, K. Kaggle competetion. *https://www.kaggle.com/competitions/trec-covid-information-retrieval* (2020).

[3] Nadkarni, P. M., Ohno-Machado, L. & Chapman, W. W. Natural language processing: an introduction. *Journal of the American Medical Informatics Association* **18**, 544–551 (2011).

[4] Domingos, P. A few useful things to know about machine learning. *Communications of the ACM* **55**, 78–87 (2012).

[5] Collobert, R. *et al.* Natural language processing (almost) from scratch. *Journal of machine learning research* **12**, 2493–2537 (2011).

[6] Ma, L. & Zhang, Y. *Using word2vec to process big text data*, 2895–2897 (IEEE, 2015).

[7] Deepu, S., Raj, P. & Rajaraajeswari, S. *A framework for text analytics using the bag of words (bow) model for prediction*, 12–13 (2016).

[8] Zhang, Y., Jin, R. & Zhou, Z.-H. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics* **1**, 43–52 (2010).

[9] Erk, K. & Padó, S. *A structured vector space model for word meaning in context*, 897–906 (2008).

[10] Wang, P. *et al.* Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing* **174**, 806–814 (2016).

[11] Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[12] Jang, B., Kim, I. & Kim, J. W. Word2vec convolutional neural networks for classification of news articles and tweets. *PloS one* **14**, e0220976 (2019).

[13] Alammar, J. The illustrated bert, elmo, and co.(how nlp cracked transfer learning)(2018) (2018).

[14] Maher, K. & Joshi, M. S. Effectiveness of different similarity measures for text classification and clustering. *International Journal of Computer Science and Information Technologies* **7**, 1715–1720 (2016).

[15] Larsen, B. & Aone, C. *Fast and effective text mining using linear-time document clustering*, 16–22 (1999).

[16] Baeza-Yates, R. & Ribeiro-Neto, R. Modern information retrieval. ny (1999).

[17] Rahutomo, F., Kitasuka, T. & Aritsugi, M. *Semantic cosine similarity* (2012).

[18] Movassagh, A. A. *et al.* Artificial neural networks training algorithm integrating invasive weed optimization with differential evolutionary model. *Journal of Ambient Intelligence and Humanized Computing* 1–9 (2021).

[19] Alzubi, O. A., Alzubi, J. A., Al-Zoubi, A., Hassonah, M. A. & Kose, U. An efficient malware detection approach with feature weighting based on harris hawks optimization. *Cluster Computing* 1–19 (2021).

[20] Alzubi, J. A., Jain, R., Singh, A., Parwekar, P. & Gupta, M. Cobert: Covid-19 question answering system using bert. *Arabian journal for science and engineering* 1–11 (2021).

[21] Alzubi, J. A. *et al.* Paraphrase identification using collaborative adversarial networks. *Journal of Intelligent & Fuzzy Systems* **39**, 1021–1032 (2020).

[22] Alzubi, J. A. *et al.* Deep image captioning using an ensemble of cnn and lstm based deep neural networks. *Journal of Intelligent & Fuzzy Systems* **40**, 5761–5769 (2021).

[23] Abdelrazek, A., Eid, Y., Gawish, E., Medhat, W. & Hassan, A. Topic modeling algorithms and applications: A survey. *Information Systems* 102131 (2022).

[24] Khadhraoui, M., Bellaaj, H., Ammar, M. B., Hamam, H. & Jmaiel, M. Survey of bert-base models for scientific text classification: Covid-19 case study. *Applied Sciences* **12**, 2891 (2022).

[25] Incitti, F., Urli, F. & Snidaro, L. Beyond word embeddings: A survey. *Information Fusion* **89**, 418–436 (2023).

[26] Jivani, A. G. *et al.* A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl* **2**, 1930–1938 (2011).

[27] Alammar, J. The illustrated word2vec. *Visualizing Machine Learning One Concept at a Time Blog* (2019).

[28] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* **26** (2013).

[29] Guthrie, D., Allison, B., Liu, W., Guthrie, L. & Wilks, Y. *A closer look at skip-gram modelling* (2006).

[30] Mohammed, M. & Omar, N. Question classification based on bloom's taxonomy cognitive domain using modified tf-idf and word2vec. *PloS one* **15**, e0230442 (2020).