LEVERAGING NEURAL NETWORK PHRASE EMBEDDING MODEL FOR QUERY REFORMULATION IN AD-HOC BIOMEDICAL INFORMATION RETRIEVAL

Amol P. Bhopale[1*], Ashish Tiwari[2]

[1,2]Department of Computer Science and Engineering,
Visvesvaraya National Institute of Technology,
Nagpur-440010, India

E-mail: amolpbhopale@gmail.com[1*] (corresponding author), at@cse.vnit.ac.in[2]

DOI: https://doi.org/10.22452/mjcs.vol34no2.2

ABSTRACT

This study presents a spark enhanced neural network phrase embedding model to leverage query representation for relevant biomedical literature retrieval. Information retrieval for clinical decision support demands high precision. In recent years, word embeddings have been evolved as a solution to such requirements. It represents vocabulary words in low-dimensional vectors in the context of their similar words; however, it is inadequate to deal with semantic phrases or multi-word units. Learning vector embeddings for phrases by maintaining word meanings is a challenging task. This study proposes a scalable phrase embedding technique to embed multi-word units into vector representations using a state-of-the-art word embedding technique, keeping both word and phrase in the same vectors space. It will enhance the effectiveness and efficiency of query language models by expanding unseen query terms and phrases for the semantically associated query terms. Embedding vectors are evaluated via a query expansion technique for ad-hoc retrieval task over two benchmark corpora viz. TREC-CDS 2014 collection with 733,138 PubMed articles and OHSUMED corpus having 348,566 articles collected from a Medline database. The results show that the proposed technique has significantly outperformed other state-of-the-art retrieval techniques.

Keywords: phrase embedding, query expansion, biomedical information retrieval, neural network, spark map-reduce

1.0 INTRODUCTION

In the past few years, word embedding has proven to be effectively beneficial in natural language processing and information retrieval tasks. Word2vec [1] and Glove [2] are continuous space embedding models, which project words into a dense low-dimensional vector space. Word embedding is a neural network-based approach to process massive amounts of text data by representing word semantics in low-dimensional vector space. Several applications including ad-hoc search, question answering, text classification, machine learning tasks, etc. have shown improvements upon integrating word embeddings; however, the effectiveness of these embeddings is hindered by the inefficiency to derive relationships between semantic phrases. Table 1 shows the comparative differences between semantic results generated by word embedding and phrase embedding models trained on the same corpus. It contains the top 15 contextually similar words and phrases with their similarity scores for a given target term.

It is observed in Table 1 that, for target word *melena*, many meaningful phrases such as- epigastric_pain, abdominal_pain, gross_hematuria, left_flank_pain, high_fever, abdominal_mass, and rectal_bleeding and for the word *cbc* meaningful phrases such as- blood_count, blood_tests, urine_analysis, blood_counts, blood_test, platelet_count, blood_picture, abnormal_liver, and blood_chemistry are extracted using phrase embedding technique; whereas, word embedding technique fails to derive such meaningful phrases. Along with these meaningful phrases, the same lists of meaningful words are derived by phrase embedding technique as generated by the word embedding technique. Phrase embedding technique is also able to derive contextually similar terms for given phrases (say for target phrase "chest_pain" similar terms or phrases derived by the model as- {dyspnea, angina, palpitations, epigastric_pain, haemoptysis, sudden_onset, angina_pectoris, etc.}), which is not possible to derive using word embedding technique. Hence enhancing the query expansion using the phrase embedding technique may lead to improve precision in ad-hoc information retrieval tasks.

Table 1: Contextually similar words/phrases generated by Phrase Embedding and Word Embedding techniques

Phrase Embedding Results			**Word Embedding Results**	
Contextually similar to 'melena'				
Word/Phrase	**Similarity Score**		**Word/Phrase**	**Similarity Score**
hematemesis	0.8129		hematemesis	0.8486
hematochezia	0.7496		hematochezia	0.7912
epigastric_pain	0.7122		melaena	0.7197
abdominal_pain	0.6872		bloody	0.6644
melaena	0.6656		haematemesis	0.6275
haematemesis	0.6568		haematochezia	0.6241
gross_hematuria	0.6369		tenesmus	0.5968
haematochezia	0.6265		bilious	0.5903
left_flank_pain	0.6237		haemoptysis	0.5898
bloody	0.6182		hepatomegaly	0.5857
high_fever	0.6143		haematuria	0.5843
abdominal_mass	0.6101		epigastric	0.5840
rectal_bleeding	0.5957		pallor	0.5823
dysuria	0.5853		vomiting	0.5724
jaundice	0.5768		lethargy	0.5716
Contextually similar to 'cbc'				
Word/Phrase	**Similarity Score**		**Word/Phrase**	**Similarity Score**
blood_count	0.5060		hemogram	0.4888
urinalysis	0.5022		wbc	0.4854
blood_tests	0.4938		fbc	0.4585
hemogram	0.4711		urinalysis	0.4363
lft	0.4626		haemogram	0.4200
wbc	0.4603		count	0.4166
lfts	0.4427		lft	0.4141
urine_analysis	0.4423		lfts	0.3942
blood_counts	0.4420		esr	0.3587
blood_test	0.4395		anamnesis	0.3463
platelet_count	0.4381		haematology	0.3436
blood_picture	0.4361		wbcs	0.3398
abnormal_liver	0.4334		normocytic	0.3370
fbc	0.4332		asat	0.3362
blood_chemistry	0.4295		counts	0.3356

Phrase embedding is a technique for enriching words or phrases with contextually meaningful words or phrases. It is observed in the biomedical literature that, authors usually refer acronyms for medical terms which could be unigram, bigram, trigrams, or even n-grams. Medical experts routinely need such literature to understand clinical cases and make judgments, treat rare syndromes, and follow the best practices. However, using short queries with acronyms over vast growing digital medical content often retrieves irrelevant material and ends up with poor precision. This motivates us to perform phrase embedding which infers the attributes of context surrounded and captures non-compositional semantics. You need to understand where it went wrong.

More precisely, this paper builds a phrase embedding model trained on a high volume bio-medical collection to expand queries for ad-hoc information retrieval. Specifically, first, the phrases are extracted using the chunking technique that identifies the boundaries of phrases at each special character or stop-word. To efficiently work with large text collections, a spark based map-reduce framework is employed. Map function splits the dataset into small groups and generates phrases; whereas the Reduce function collects those phrases and explicitly annotates them in the dataset. We then learn phrase embedding over the tagged data collection, which treats each phrase as a single unit of a word and generates an embedding model same as generated by the word embedding technique. These phrase embeddings are further used in query reformulation and expansion. Query expansion is performed by extracting contextually similar terms or phrases for a given query term using the trained embedding model. The effectiveness of the proposed technique is measured by retrieving relevant documents for a set of queries over two bio-medical corpora. The study has also shown the comparison with state-of-the-art retrieval models i.e. with- simple IR without query expansion model, word embedding based query expansion model, and pseudo relevance feedback based query expansion model. The key contribution of this paper involves:

- Employed chunking technique to identify the phrase boundary at every stop-word. We set-up a spark map-reduce environment for a large dataset and perform phrase extraction and corpus annotation.
- Phrase embedding is learned on the newly annotated corpus using word2vec continuous bag-of-words (CBOW) model [1] that considered phrases as a single-unit.
- The proposed Phrase embedding model is used to expand query terms with its contextually similar words or phrases.
- To evaluate the performance of the proposed approach, we conducted exhaustive experiments on two different medical test collections (TREC-CDS 2014, OHSUMED)
- Exhaustive set of experiments are carried out to compare and analyze the results with state-of-the-art retrieval models.

The remainder of this paper is organized as below- section 2 presents the detailed theoretical overview on- word vector representation, distributed representation of terms or phrases i.e. word embeddings and phrase embeddings, and also their applications in query expansion task. Section 3 provides details about the proposed framework. Experimental setup, results, and comparative analysis with state-of-the-art retrieval models are presented in section 4, followed by a conclusion and opportunities for future work in section 5.

2.0 RELATED WORK

There have been significant efforts taken by researchers in developing different IR models such as- Boolean and Boolean variant models, Vector space model, Linguistic model, and Probabilistic models. In recent years embedding techniques have accelerated the improvements in NLP and IR related tasks. Although embedding is not a new concept; it was first introduced in 2003 by Bengio et al. [3] who developed a neural probabilistic language model by learning the distributed representation of words to address the curse of dimensionality caused by one-hot vectors. However, the benefits of distributed representation of words i.e. word embeddings as a characteristic of NLP tasks were first shown in [4]. This study briefly surveys previous work specifically applied for word representation learning using different techniques such as- vector space model, word embedding, and phrase embedding techniques. It also presents studies that apply these word representation learnings for query expansion task of IR.

Vector Space Model (VSM)
VSM is one of the simplest techniques to represent data in an algebraic format. It was first devised in 1975 by G. Salton et al [5]; they built document vectors having word vocabulary as dimensions or features. For each vector, weights of dimensions are computed using the frequency of words in a document; however, different weighting techniques have been explored, mainly based on frequencies and normalized frequency [6]. Document-based VSM is also extended for other small units such as- words. Here words are generally considered as a point in vector space and these vectors are represented by calculating frequencies of co-occurrences of words in the document collection [7]. Document-based VSM and word-

based VSM techniques have successfully been used for many years in various text processing techniques such as- information retrieval, information extraction, text classification, document clustering, sentiment analysis, word sense disambiguation, etc. Despite having many applications, VSM based techniques have the curse of dimensionality, since vocabulary words act as a dimension which easily could grow to thousands or millions for larger datasets. An important dimensionality reduction technique adopted in many studies using singular value decomposition known as Latent Semantic Analysis (LSA) [8] in which document vectors are represented in word space by considering inter-word dependencies. The only drawback of these techniques is that for modeling word associations they considered word co-occurrences at the document level. In recent years neural network-based unsupervised models are studied to learn low-dimensional word representations.

Word Embedding

Word embedding is the neural network approach that learns low-dimensional word representations by considering the local context i.e. window-based context words around the target word. It enables to identify the similar context word used for the given target word and captures these similarities at the semantic level with better compositionality. As discussed earlier embedding is not a new concept; however, researchers started using it widely after 2013, when T. Mikolov et al. [1] introduced word2vec, an open-source embedding model for word representation. Word2vec has two different but related models: Skip-gram and Continuous Bag-of-Word (CBOW). CBOW framework is based on the feed-forward neural network language model [3] which predicts the target word using surrounding context words traced by a moving window. The architecture of the Skip-gram model is exactly similar to CBOW, but it predicts surrounding words for a given target word. Variants of word2vec such as- Glove [2], WordRank [9], FastText [10], Paragraph2Vec [11] etc. are introduced in later time.

In the context of IR, Word Embedding has been widely used as a solution for vocabulary mismatch problem by generating semantically similar terms. In [12] authors present a distributed semantics representation i.e. embedding-based term reweighting technique used for language modeling and BM25 retrieval models. Mitra et al [13] proposed a hybrid compositional model with word embedding to represent queries and documents; and used the cosine similarity function to rank documents. This ranking model helps to improve the performance of the IR model. A cosine similarity based probabilistic translation language model is derived using word embedding in [14, 15]. Bilingual word embedding models are learned using document-aligned comparable datasets and are applied for IR [16].

Phrase Embedding

An integral limitation of word vector representations is their incapability to represent phrases. The best-known work done even before word2vec became popular was in [17, 18], where constituent word vector representations are used to calculate distributed representations of a particular phrase or a sentence. Inspired from the word embedding techniques, multiple units of continuous and non-contiguous words, i.e. phrases are derived using statistical approaches and considered them as a single unit to represent in distributed vectors [19]. In [20] authors have proposed a feature-rich compositional transformation (FCT) model for phrase vector representation. The vectors are computed using the weighted sum of word vectors based on the feature list for each phrase. A hybrid method based on a linear combination of the distributional and compositional components with compositionality constraints for each phrase was built to learn phrase embeddings [21]. This study first extracts the phrases and learns distributed representation using the word2vec CBOW model by considering phrase as a single unit.

Query Expansion

User-specific queries are too short to accurately explicate the information requirements; hence, to improve the retrieval effectiveness queries are expanded by adding contextually similar terms. There are number of query expansion approaches studied in the past, however, the scope of this study is limited to focus on linguistic approaches. In the paper [22], authors have proposed a statistical linguistic technique for ambiguous query reformulation. Here, the query is first normalized to identify concepts using a knowledge base and stored it in the container which is further used by a system to match between noun query tokens. For the ambiguous query whose concept couldn't be identified, they applied a query expansion technique using WordNet. K-nearest neighbor technique is applied in [23] to select similar terms from the embedding vectors. Query terms are ranked by calculating the average cosine similarity for each neighboring term w.r.t all query terms. The main drawback of the K-NN technique is to consider similar words which are at a short distance. It is overcome in [24] by selecting similar terms based on a global threshold value. This threshold value is calculated by taking the average number of synonyms over all words in the corpus. In[25] authors have proposed an embedding vector-based heuristic

method VEXP to expand query terms by adding most similar terms of the embedding space. They defined tuning parameters and considered frequencies while weighing the terms in the expanded query.

3.0 PROPOSED FRAMEWORK

This study develops a generalized embedding model with an intention to train a system to predict similar words and phrases over a large biomedical dataset. In this work, a Spark-based map-reduce framework is applied for phrase extraction and dataset annotation. In the newly annotated dataset, the derived phrase is treated as a single non-divisible unit and using a word2vec CBOW approach the model is trained to learn the phrase and word embeddings. This embedding helps to derive semantic relationships amongst words and phrases. A query expansion model is built-up using these embeddings, and documents are retrieved using expanded queries. Fig. 1 shows the framework of the proposed approach.

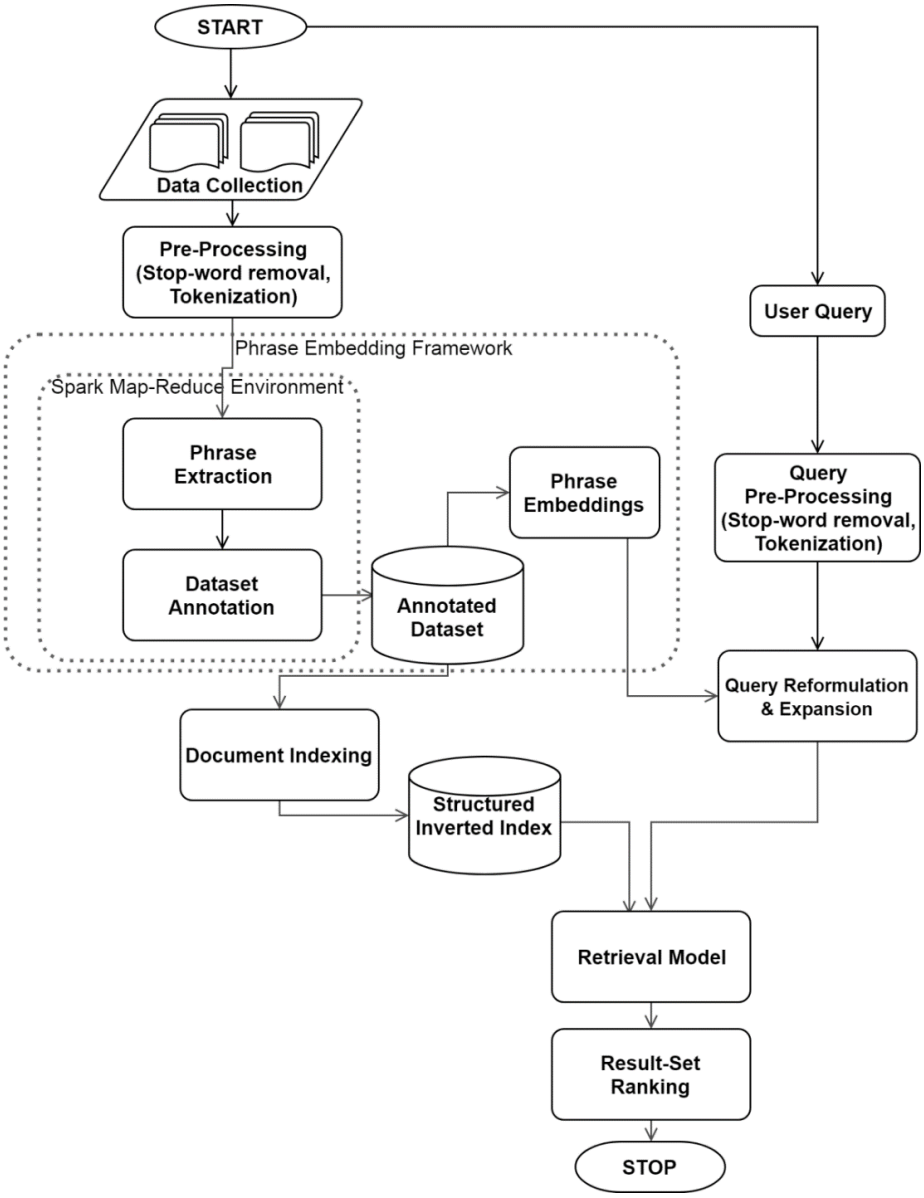


Fig. 1: Framework for the proposed Approach

3.1 The Phrase Embedding Model

As shown in fig. 2, the phrase embedding process is divided into three stages. In the first stage, frequent phrases are extracted from the corpus. In the second stage, extracted phrases are tagged to generate a new annotated dataset. In the third stage, the newly built annotated corpus is used to train the generalized embedding model using the word2vec CBOW technique. Each of these stages is discussed in the following subsection.

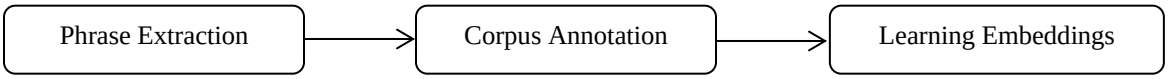


Fig. 2: Three Phase Proposed Approach

3.1.1 Phrase Extraction

To capture continuous and non-contiguous phrases, [26] proposes a scoring technique based on the co-occurrences of word frequencies with a discounting parameter to penalize rare words; however, this technique is ineffective to extract rare n-grams and hence end-up with very limited selective phrases. We employed a chunking technique proposed by Abney [27] in 1991. It is a linguistically heavy approach where the entire corpus is split-up into segments at every special character such as comma, semicolon, period, etc. Segments are tokenized and boundaries for candidate phrases are fixed at occurrences of every stop-word. Rare candidate phrases are filtered out based on the threshold frequency count. Fig. 3 briefly shows the steps followed in phrase extraction.

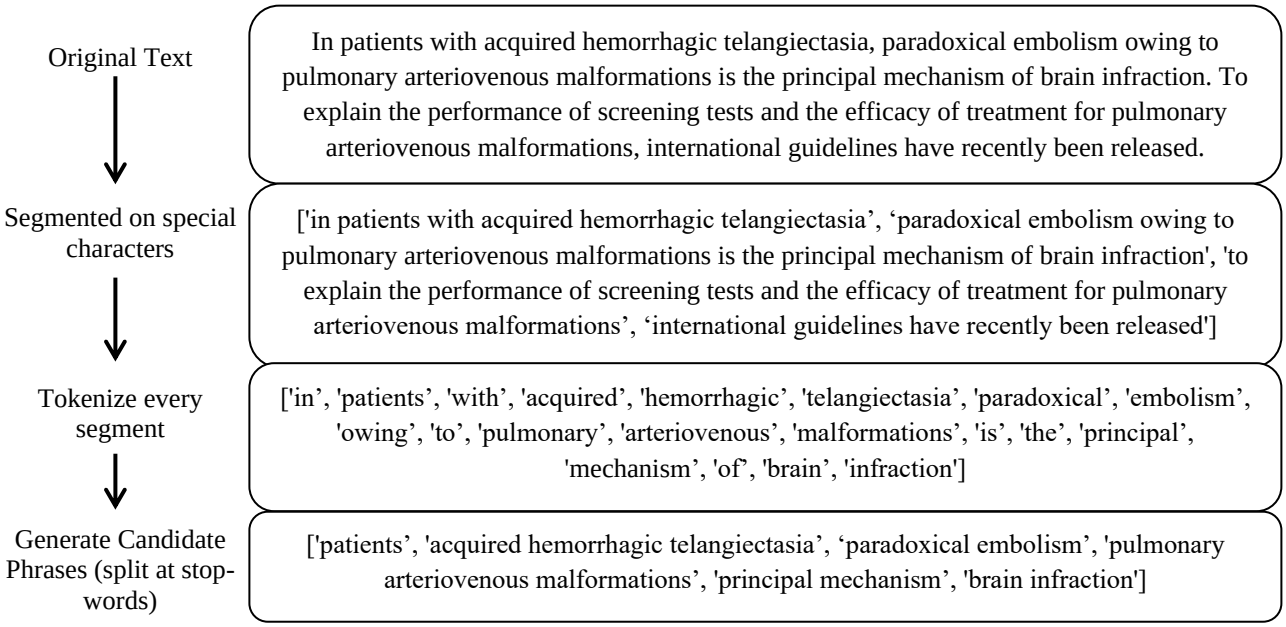


Fig. 3: Phrase Extraction Sequence

This phrase extraction process is enhanced using spark map-reduce environment to efficiently apply on large scale datasets. As shown in fig 4. the *map* function splits the dataset into small parts and applies the above discussed NLP techniques to generate candidate phrases, whereas the *reduce* function collects phrases that satisfy the minimum threshold count and store it for future use. Table 2 shows the sample phrases extracted from the corpus, where it contains bigrams, trigrams and in some cases higher-order n-grams also.

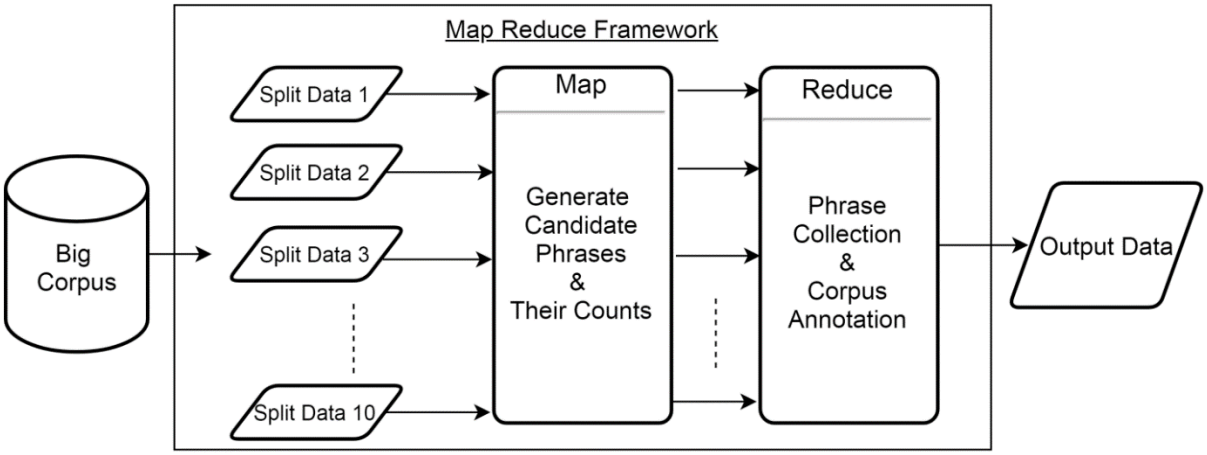


Fig. 4: Map Reduce Framework

Table 2: Sample candidate phrases extracted from corpus.

gene associated	patient treated	prospectively included
review attempts	measles virus	stochastic model
kidney diseases	therapeutic agent	cold acclimation
neural basis	iron homeostasis	ocean acidification
radiation exposure	risk assessment	detailed review
spinal surgery	investigate effects	personality traits
proteolytic cleavage	specific differences	gren syndrome
major goals	deep venous thrombosis	cardiac arrest
optimal level	different processes	synapse formation
highest affinity	large parts	clinical consequences
showed improvement	large range	differentially regulated genes
functional networks	tumor regression	obsessive compulsive disorder
rheumatoid arthritis	warranted conclusion	retrospectively reviewed patients
clinically suspected	chemical carcinogens	inter xadmolecular inter xadactions

3.1.2 Corpus Annotation

In this stage, the derived phrases are tagged in the corpus to consider it as a single non-divisible unit in the embedding process. This can be done in two ways, first by annotating the entire corpus to build a new annotated version; or a separate layer needs to be added in a neural network to annotate each document with phrases before learning embeddings. This study implements the first approach and as shown in fig 4. the *reduce* function is used to build a new pre-annotated corpus wherein each derived n-gram white-spaces in between are replaced by the underscore character. Fig. 5. shows a sample document in the newly annotated corpus.

> In patients with acquired_hemorrhagic_telangiectasia, paradoxical_embolism owing to pulmonary_arteriovenous_malformations is the principal_mechanism of brain_infraction. To explain the performance of screening_tests and the efficacy of treatment for pulmonary_arteriovenous_malformations, international_guidelines have recently been released.

Fig. 5: Sample annotated document in a corpus

3.1.3 Learning Phrase Embedding

Further, we propose to use the word2vec CBOW model [1] to train vector representations of phrases or terms and learn phrase embeddings on an explicitly tagged corpus. The main idea behind the proposed approach is to treat multiple units of the word as a single unit and represent its vector in the semantic space of other vectors. CBOW model helps to learn the embeddings i.e. it generates the probabilities at the output layer by considering the surrounding context and predicts the target word by iteratively working on minimizing the loss function. As discussed earlier, the CBOW framework is based on the feed-forward neural network language model which works in three different identifiable layers- an input layer, hidden layer, and output layer as shown in fig 6.

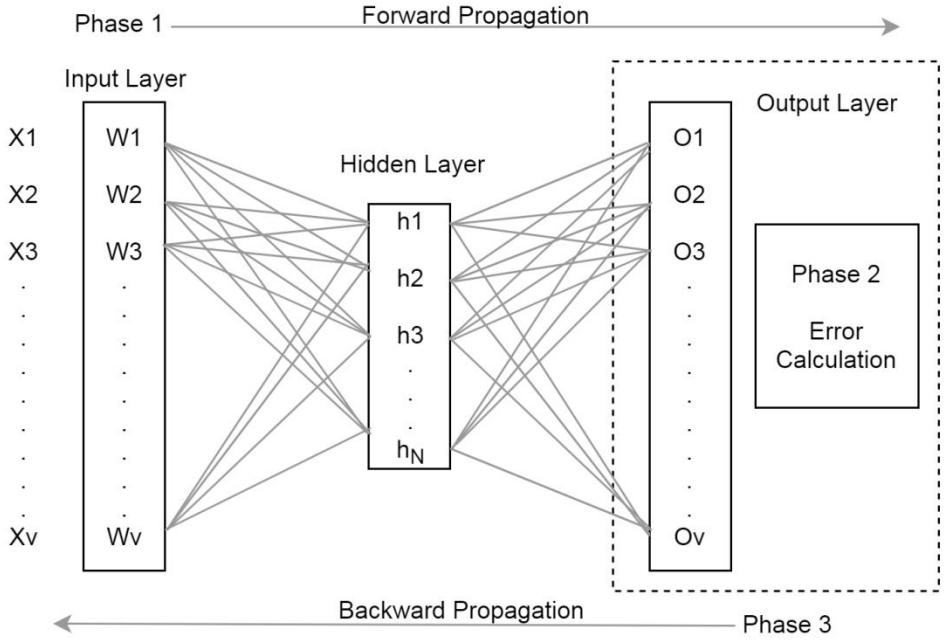


Fig. 6: Single Context CBOW model

The input layer encodes with the one-hot vector of vocabulary size and the number of neurons present in the hidden layer depends on the dimensions of the vector. The number of nodes present in the output layers is the same as in the input layer. The Neural network model follows three steps to learn embeddings- forward propagation, error calculation, and backward propagation.

Forward Propagation
In forward propagation as shown in fig 6. the input layer is fed-up with a one-hot vector of the context i.e. with a binary vector and has value 1 for each context word considered while keeping the others as zero. Let W be the initial weight of a link between the input layer node and the hidden layer node. The Weight of the neurons in the hidden layer is calculated without considering the activation function as shown in equation (1)-

$$h_i = \sum_{j=1}^{V} W_{ji} * X_j \quad (1)$$

Where, X_j is the input parameter, W_{ji} is the initial weight between input node X_i and the hidden layer node h_i and V is the vocabulary size.

In the next part, we calculate the weights of nodes in the output layer. It includes both association and activation function (a softmax function). For each output node, the association function is calculated as the sum of the product of h_i and the corresponding weights between hidden to output layer node; given as below-

$$u_i = \sum_{j=1}^{N} W'_{ji} * h_j \quad (2)$$

Where, hj is the weight of a hidden layer node calculated in equation (1), W'_{ij} is the initial weight between hidden layer node h_j and the output layer node O_i, N is the number of neurons present in the hidden layer.

The softmax function is used to calculate the activation scores of nodes in the output layer, hence the softmax output for the word w_t w.r.t $w_c = \{w_{t-n},..,w_{t-1}, w_{t+1},.., w_{t+n}\}$, a given sequence of context words with window size $ws = 2n$ can be given by the conditional probability in terms of exponentials as-

$$P(w_t, w_c) = \frac{e^{u_t}}{\sum_{t'=1}^{V} e^{u'_t}} \quad (3)$$

Hence the objective function is to maximize the probability of observed word-context pairs which is given by the equation (4).

$$y_j = \frac{1}{V}\sum_{t=1}^{V} \log p(w_t | \sum_{-c \le j \le c; j \neq 0} w_{t+j}) \quad (4)$$

Where c is the context window size, w_t is the target word and V is vocabulary size.

Error Calculation
In the error calculation stage, a difference is calculated between the actual word vector and the summed-up values at the output layer. As discussed above, the training objective is to maximize the conditional probability of observing actual word w_t for the given input context w_c, thus the loss function is defined as-

$$\max (P(w_t|w_c)) = \max (y_j) \quad (5)$$

This loss function can also be considered as a special case of cross-entropy measurement between two probabilistic distributions, and the error is calculated as-

$$E(O_t) = log \frac{e^{u_t}}{\sum_{t'=1}^{V} e^{u'_t}} \quad (6)$$

Where O_t is the corresponding node in the output layer.

To minimize the error we need to negate the above equation (6), therefore the required minimum error E is given as below-

$$E = -log\ P(w_t|w_c) \quad (7)$$

Backpropagation
In backpropagation, a gradient descent optimization algorithm is applied to update the weights of all links between neurons at the output and hidden layer; and links between neurons at the hidden and input layer. The weighted matrix between the input layer and the hidden layer is essential to be considered as a word vector being learned. The i^{th} row of the matrix gives the n-dimensional embedded vector for word w_i.

All three steps, i.e. forward propagation, error calculation, and backpropagation are iteratively performed until we get optimal weights at all neurons. Once this model is trained on a large corpus, it can be used to predict words for the given context. The generalized CBOW model with C context words $\{X_1,X_2,\ldots,X_C\}$is shown in fig 7.

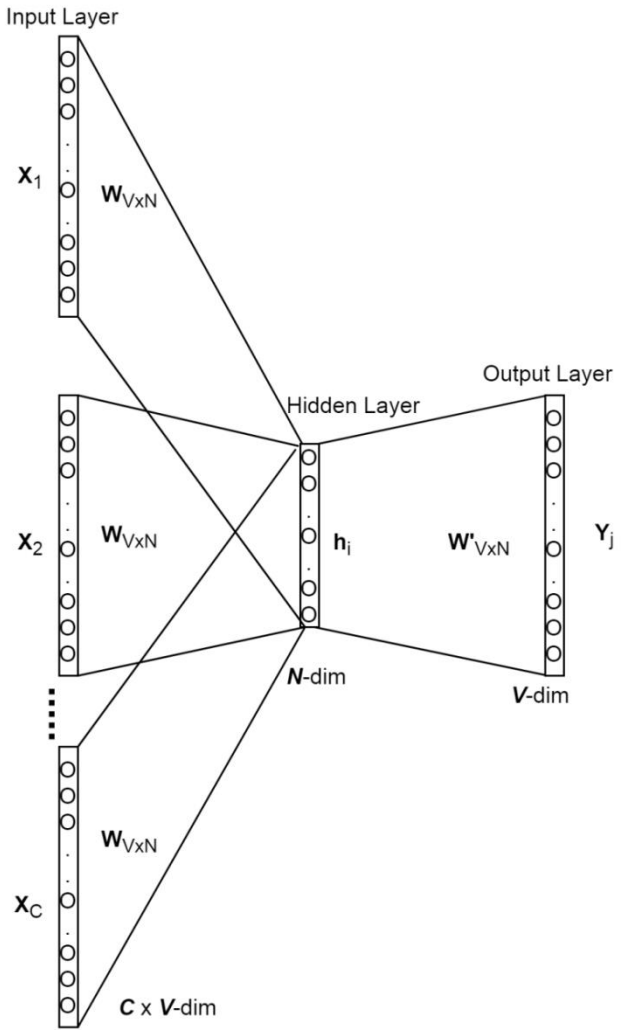


Fig. 7: Generalized CBOW model

3.2 The Query Expansion Model

In language models, query expansion involves estimating an alternative expanded language model which is then interpolated with the original query model. As discussed in the previous section, we used a neural network model for vector representation and learned semantic relationships between different phrases and terms. Therefore, we represent each term t into a vector v_t of predefined dimension, say 300. For query expansion, we used these learned relationships and applied normalized cosine similarity measure on vectors to extract similar phrases or terms. The measure is defined as below-

$$SIM(t_1, t_2) = \widetilde{cos}(v_{t1}, v_{t2}) \quad (8)$$

Where, the similarity value ranges from 0 to 1. We prepared a list *top-k(q)* of k most similar terms or phrases as an expansion of the given query term. In the following, we explain the building of query expansion model.

Let a query Q be represented as a set of different terms (eg: Q = "*history of smoking, heavy drinking, with dysphagia*"). We applied pre-processing techniques such as stop-word removal and tokenization and also removed duplicate terms to obtain *m* unique query terms i.e. Q – {q_1, q_2, q_3, q_4,..,q_m}(eg: after pre-processing, Q = "*smoking, drinking, dysphagia*"). We followed the following procedure to expand each query term q_i:

- For a term $q_i \in Q$ (say for the term '*drinking*'), take *k-most* similar terms or phrases from the list (we experimented with $k = 1, 3, 5, 7, 9$); for the term '*drinking*' with $k = 3$ i.e. top 3 similar terms extracted using a phrase embedding model are- "*heavy drinking, binge drinking, alcohol*".
- To have more relevant documents with the original query term *q*, assign higher weights to *q* than the terms used for expansion i.e. we doubled the weight of original query term than the *top-k* terms.
- We reformulated expanded query using Boolean combinations of AND and OR; such that, more correct results should be retrieved with possible combinations of query terms (eg: *"smoking OR drinking"; "smoking AND drinking"*).
- A query term q_i is expanded as below:

  $$q_i' = q_i^2 \bigcup_{q \in Q} (top - k\ (q))$$

  Here, the q_i^2 represents the more weightage, given to the original query term than the *top-k* similar terms (eg: the term "*drinking*" is expanded as "*drinking*^{2}, *heavy drinking, binge drinking, alcohol"*).
- A complete expanded query can be re-written as below:
  Q' = Boolean(q_i') for $i = 1$ to m.
  (eg: *Q' = "(smoking*^{2}*, tobacco_smoking, smokers, current_smoking, drinking*^{2}*,heavy_drinking, binge_drinking, alcohol) AND (dysphagia*^{2}*, odynophagia, swallowing, hoarseness)")*

Thus, the derived expanded query *Q'* is used in retrieval models to generate relevant results.

3.3 Document Retrieval Architecture

The retrieval process is highly affected by practices followed in storing and mapping the content. In this work, an inverted index technique is applied which allows indexing by mapping vocabulary words to the list of documents in which they appear. Before actual indexing, we removed stop words to have noise-free content. Inverted indexing is a simple data structure and allows fast accessing of full-text documents from large data collections.

Query matching results are ranked by computing scores using various term-weighting schemes such as term frequency inverse document frequency (*TfIdf*) [28], *Cosine* [29], *BM25* [30], etc. It is observed that the ranking functions usually favor longer documents over the shorter one; hence to correctly rank these results, length normalization techniques are incorporated while generating scores. In paper [31] a comparative analysis of different ranking techniques is discussed and suggested that the *BM25* technique generates good results over other scoring techniques. In the remaining part of this section, this study presents some of the widely used scoring and ranking techniques and applies it with the proposed phrase embedding model.

3.3.1 TFIDF

TfIdf is a widely used term weighting scheme that assigns weights to terms based on their occurrences in a document and the entire corpus. For the term, which is frequently occurred in a document and rarely appeared in the entire corpus; this function generates a high score. Thus, indicating that the term carries more information. *TfIdf* is defined as-

$$TfIdf(t,d) = tf(t,d) * \log\left(\frac{N}{df}\right) \tag{9}$$

Where, *tf(t,d)*: term frequency count of term *t* in document *d*, *N*: Total number of documents in a corpus, *df*: document frequency i.e. number of documents where term *t* occurs.

3.3.2 Cosine

It is a measure of the orientation of two vectors, which generate scores of similarity without considering the vector magnitude. If two document vectors have the same orientation, then the cosine measure produces score 1; if vectors are orthogonal it generates 0 and for opposite vectors, -1 is assigned. The cosine similarity measure is generally used in positive space and value ranges from 0 to 1. The formula is derived from the Euclidean dot product, where the score is generally calculated as the ratio of the dot products of two vectors to its magnitude. In this work, a cosine angle between normalized document vector *D* and query vector *Q* is measured, which has n different components i.e. the dimension size of a vector. Cosine measure is given as in the below equation (10).

$$\cos(\theta_{D,Q}) = \frac{\sum_{i=1}^{n} D_i * Q_i}{\sqrt{\sum_{i=1}^{n} D_i^2}\ \sqrt{\sum_{i=1}^{n} Q_i^2}} \tag{10}$$

3.3.3 BM25 and BM25F

BM25 is a well-known document scoring function based on the probabilistic retrieval technique mostly used in search engines to rank results set generated for queries according to their relevance. It is also known as Okapi *BM25*, where *BM* stands for best matching. It is an iteration on the state-of-the-art *TfIdf* technique with several modifications. *BM25* ranks result set based on the query terms appearing in each document. *BM25* generates a score for a document *D* w.r.t. the given query *Q* containing terms $q_1 \ldots q_n$, as given in equation (11):

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i).\ \frac{f(q_i,D).(k_1+1)}{f(q_i,D) + k_1.(1 - b + b.\frac{|D|}{avgdl})} \tag{11}$$

Where, *n* : number of query terms; $f(q_i;D)$: query term frequency in document *D*; |D|: length of document calculated by number of words it contains; *avgdl* : average document length in the corpus; k_1, *b*: are tuning parameters used for smoothing correction and document length normalization with varying values as- k_1 = 1.2-2.0, *b* = 0:75; *IDF*: inverse document frequency of query term q_i given in below equation (12):

$$IDF(q_i) = log\ \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \tag{12}$$

Where, *N*: total number of documents in a corpus; and $n(q_i)$: number of documents containing q_i.

BM25 works well with shorter queries and unstructured documents, whereas the variant of *BM25* i.e. *BM25F* assigns a different degree of importance to various fields in a document i.e. if the document has fields like title, abstract, keywords, description etc. then *BM25F* score is calculated as in below equation (13):

$$BM25F = IDF(q_i) * \left(BM25_{title} + BM25_{abstract} + BM25_{keywords} + BM25_{description} + \cdots\right) \tag{13}$$

Where $IDF(q_i)$ is same as given in equation (12). *BM25* and *BM25F* techniques have been used in various retrieval models to improve relevance scoring.

4.0 RESULTS AND DISCUSSION

This section discusses the details of the experimental setup, which briefly presents the datasets used for experimentation, the performance evaluation measures, and the setting of the parameters. In the later part of this section, we present the experimental results and analysis.

4.1 Experimental Setup

All experiments are carried out on Intel(R) Xeon processor with RAM size equals 32GB and used python as a coding language to implement all modules. Whoosh[1], an open-source search engine purely implemented in python is used to build the document index. To evaluate the performance of the proposed system, experiments are conducted on TREC-CDS 2014 and OHSUMED data collections, and the results are compared with state-of-the-art techniques.

4.1.1 Dataset Preparation

For experimentation and performance evaluation, we used two widely known biomedical article's datasets.

[1] https://pypi.org/project/Whoosh/

- *TREC-Clinical Decision Support 2014 Dataset*
  TREC-CDS dataset is widely used in many IR tasks. It has 733,138 biomedical artifacts in NXML format, collected from the PubMed Central repository. These documents are parsed using an XML parser to prepare text files having title, abstract, and keyword fields. TREC-CDS 2014 track has provided 30 queries along with their relevance records.

- *OHSUMED Dataset*
  OHSUMED document collection is prepared for the TREC-9 track which contains the title, abstract, and MESH indexing terms from 270 medical journals collected for the period of 5 years from 1987 to 1991. It has 348,566 records collected from MedLine, an online medical information repository. TREC-9 track has also provided 63 queries along with their ground truth records prepared by experts.

Table 3 gives the brief details about dataset statistics along with a number of phrases extracted and the unique tokens considered to train our embedding models

Table 3: Basic Statistics of OHSUMED and TREC-CDS 2014 Dataset

Dataset Name	OHSUMED	TREC-CDS 2014
Document Type	Medline Articles	PubMed Articles
Document Count	348,566	733,138
Query Count	63	30
Vocabulary	50,778	162,876
No. of Phrases Extracted	13,968	39,654

4.1.2 Performance Evaluation Measures

This study has used the *trec_eval* script to measure the performance of the proposed work. It considers some of the widely known evaluation techniques such as precision@k, mean average precision (MAP), and Rprec. These measures are applied to records retrieved by the proposed model against the gold standard query relevant document files prepared by experts for each dataset. The following are the details of each metric.

- *Precision (P)* This metric measures the capability of the system to generate relevant records, it estimates value based on how well the system eliminates unwanted documents. It is the ratio of relevant documents observed to the total retrieved documents and given as below-

$$P = \frac{\#\ Relevant\ Retrieved}{\#\ Total\ Retrieved} \tag{14}$$

  Precision is further measured at a different number of retrieved records i.e. *k*. We measured precision as *k*= 5, 10 i.e. *P*@5 and *P*@10, which gives a relevant count at first 5 retrieved records and first 10 retrieved records respectively.

- *Recall (R)* Recall measures the correctness of a system; it tracks how well a system retrieves what users want. It is the fraction of relevant records retrieved to the total number of relevant records.

$$R = \frac{\#\ Relevant\ Retrieved}{\#\ Total\ Relevant\ in\ Corpus} \tag{15}$$

- *R-prec* This metric generates the precision value for any specific query after retrieving R relevant records. Generally, it is observed that both the precision and recall equals at the Rth position. Often, R-prec and MAP are highly correlated to each other. R-prec is defined as-

$$R - prec = \frac{|r|}{R} \tag{16}$$

Where, $|r|$ - is the number of relevant documents retrieved amongst the top R documents. R is the total number of relevant documents for any specific query q.

- *Mean Average Precision (MAP)* Often precision and recall are considered to be single value measures and generate values based on the complete document list returned by the system. For ranking systems, it is desirable to consider the order in which the system has returned documents. MAP is a more powerful and stable metric calculated combinedely using both precision and recall. It provides a single numeric value at different recall levels for the whole system. MAP is the mean of average precision scores calculated for each query.

  For a query q_i from the set of queries Q, let R_i be the relevant documents retrieved by the system. Let $P(R_i[k])$ is the precision calculated until $R_i[k]$ is observed in the ranking. If the k^{th}-ranked document is not retrieved, then $R_i[k] = 0$ for that k value.

  Hence the average precision score for query q_i is calculated as-

$$AvgP_i = \frac{1}{|R_i|}\sum_{k=1}^{|R_i|} P(R_i[k]) \tag{17}$$

  And the mean average precision is given as-

$$MAP = \frac{1}{|Q|}\sum_{i=1} |Q_i| AvgP_i \tag{18}$$

4.1.3 Parameters setting for the proposed approach

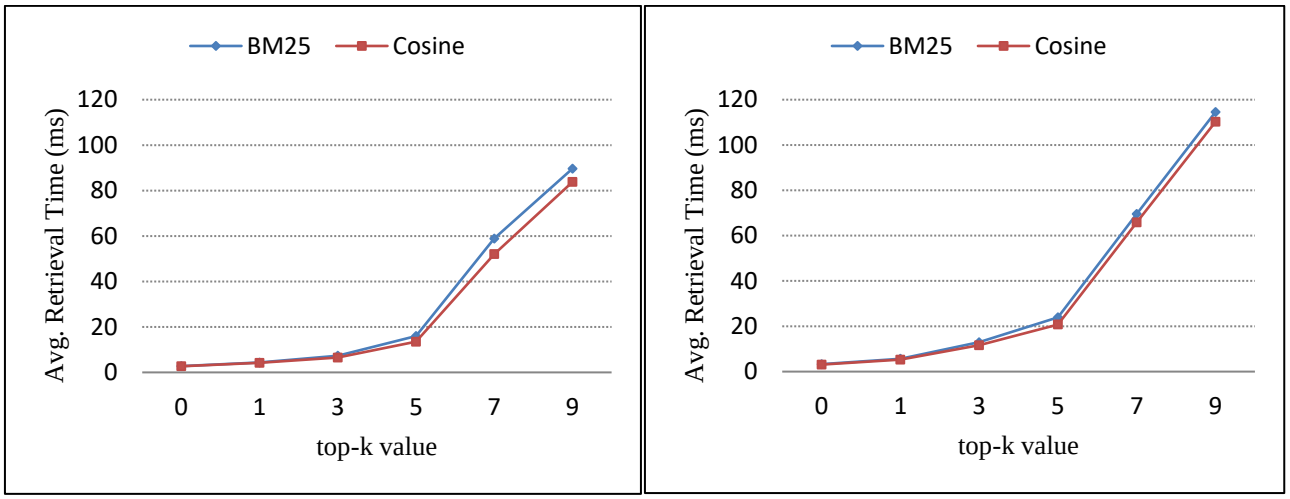
As shown in fig. 1, the first step in the proposed work is to extract phrases and tag it in a dataset. To efficiently work with large datasets and to have a scalable solution, the spark-based map-reduce framework is employed. Dataset is partitioned into 10 subparts using 10 threads and different NLP techniques are applied to remove noise from the documents. The boundary of phrases is marked at every occurrence of stop-word. Phrases with a frequency count greater than 50 for TREC-CDS 2014 and 30 for OHSUMED dataset, are collected by the *reduce* function. Further, these phrases are used to annotate the dataset.

The CBOW technique, which is implemented in the Python gensim library, is used to learn phrase embedding. The vector size, i.e. the dimension, is set to 300 with a context window of size 10. This means that a maximum of 5 context words are considered on each side for a target word to be predicted. When learning the model, the word counts less than 10 are ignored. To train the embedding model, 10 iterations are performed using 20 threads.

This study used the trained phrase embedding model to extract top-k contextually similar words or phrases for query expansion. To fix the value of *top-k*, several experiments are conducted and observed the mean average precision score generated by BM25 and Cosine retrieval model with different *top-k* values. It is evident from table 4 that the MAP score generated has a higher value for *top-k = 3*. The reason behind decreasing MAP for higher values of *top-k* is the expansion of query terms with relatively less contextually similar words in the list. For *top-k = 1*, results are poor because only one similar term is used for query expansion, which is sometimes not sufficient. It is shown in fig. 8 that, retrieval time increases with increasing *top-k* value. This is because the retrieval system takes more time to match all terms in the expanded query, hence more the number of terms present in a query more will be the retrieval time. Thus, the *top-k* value is heuristically fixed to 3 and performed the rest of the experiments. Table 5 presents a summary of the parameters used with their values and explain each of them briefly.

Table 4. Mean average precision score generated for different top-k values

	OHSUMED		TREC-CDS 2014	
Top K terms	**BM25**	**Cosine**	**BM25**	**Cosine**
0	0.1765	0.1591	0.1559	0.1320
1	0.2272	0.2115	0.2143	0.2091
3	**0.3095**	**0.2800**	**0.2899**	**0.2747**
5	0.3089	0.2802	0.2867	0.2696
7	0.2512	0.2447	0.2413	0.2383
9	0.2038	0.1998	0.2101	0.2049



(a) OHSUMED Dataset (b) TREC- CDS 2014 Dataset

Fig. 8: Effect of varying top-k value on Average Retrieval Time taken by a system in milliseconds to retrieve top 100 documents.

Table 5: Summary of parameters and their values used for experimentation

Parameter	**Description**	**Value**
Spark Threads	No. of threads used by map-reduce function	10
Min Phrase Count	Minimum frequency count of phrases to be extracted for corpus annotation, respectively for TREC-CDS and OHSUMED datasets	50 , 30
Vector Dimension	It is the length of vector considered for learning embedding model	300
Context Window Size	Number of context words to be considered for generating embedding model for a given target word	10
min_count	Minimum frequency count of words considered to ignore during learning embeddings	10
No. of Epochs	It gives the total number of forward and backward propagation needs to be performed on corpus during model training	10
CBOW Threads	It is the number of threads used for model training	20
top-k terms	It is a number of contextually similar terms used in query expansion for each query terms. Its value is required to be fixed heuristically.	3

4.2 Experimental Results

Results are generated by executing the proposed framework's pipelined tasks with the parameters tuned as given in table 5 on two different datasets. The performance of the scalable framework is validated and compared against the sequential framework. From table 6 it is observed that in a sequential environment phrase extraction task takes 4X to 7X more time and the corpus annotation task takes 12X to 13X more time than the applied map-reduce technique. Hence, employing a sequential framework on larger datasets with millions of documents is not feasible. Table 6 gives the different computing times taken by each individual module of the phrase embedding technique and a table entry NA indicates that the particular task is independent of the map-reduce environment. Hence, the time taken by these tasks is the same in both sequential and distributed environment.

Table 6: Time taken by different modules of Phrase Embedding; 'NA' indicates tasks are independent of map-reduce environment hence the same time will be taken by phrase embedding learning and document indexing tasks (all timings are in minutes)

	Map-Reduce Framework		**Sequential Framework**	
Tasks	**OHSUMED**	**TREC-CDS 2014**	**OHSUMED**	**TREC-CDS 2014**
Phrase Extraction	1.31	4.06	9.23	18.48
Corpus Annotation	12.21	77.38	167.36	937.28
Phrase Embedding Learning	11.48	37.32	NA	NA
Document Indexing	57.29	124.46	NA	NA

Table 7 gives the experimental results obtained using the proposed approach. It is observed that the BM25 weighting model has generated better retrieval results compared to other weighting schemes. BM25 weighting technique often achieves better performance, as it considers term frequency, inverse document frequency, and the average document length with some heuristic parameters. We have also mentioned an average retrieval time taken by our retrieval model to bring the top 100 matching documents for each query. Although the BM25 technique is slightly slower than other weighting schemes, but produces higher precision values which are helpful in making clinical decisions. It is observed that the retrieval time increases with the corpus size. Hence retrieval on TREC-CDS took more time than the OHSUMED dataset.

Table 7: Quality of returned documents in terms of P@5, P@10, MAP, and Rprec metrics by using phrase embedding query expansion technique (ART is an average retrieval time in milliseconds taken by a model to retrieve top 100 documents)

Dataset	**Scoring Function**	**P@5**	**P@10**	**MAP**	**Rprec**	**ART/Query @100 in ms**
OHSUMED	**Term Frequency [28]**	0.4349	0.4349	0.2511	0.4629	5.9960
	TFIDF [28]	0.5016	0.5079	0.2989	0.5211	6.6077
	Cosine [29]	0.4794	0.4794	0.2800	0.4945	6.5272
	BM25 [30]	**0.5079**	**0.5175**	**0.3095**	**0.5342**	**7.3197**
TREC-CDS 2014	**Term Frequency [28]**	0.4867	0.4867	0.2467	0.4878	10.6855
	TFIDF [28]	0.5268	0.5367	0.2701	0.5232	11.8881
	Cosine [29]	0.4535	0.4968	0.2747	0.5263	11.6249
	BM25 [30]	**0.5133**	**0.5333**	**0.2899**	**0.5416**	**12.9875**

4.3 Comparison with state-of-the-art

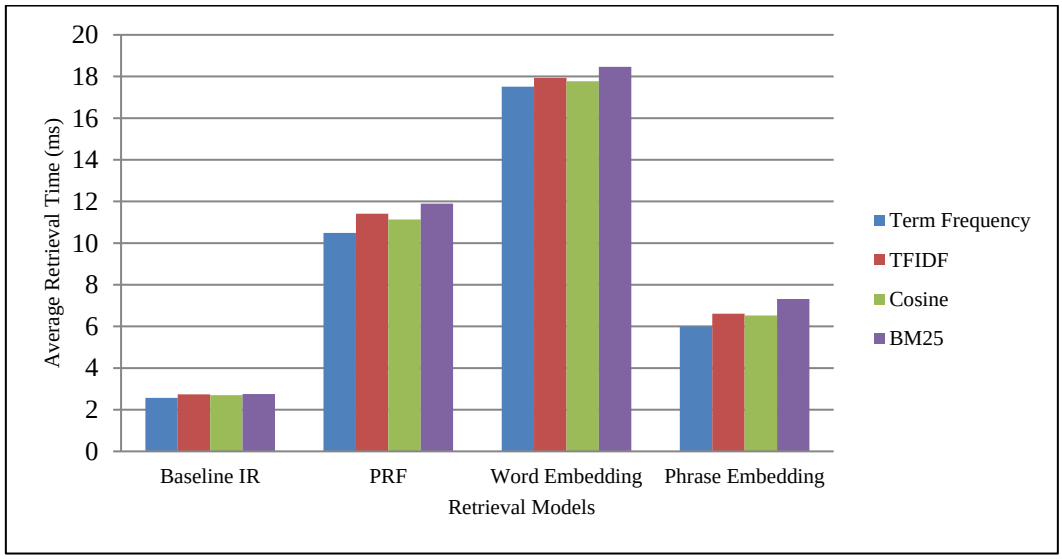
The performance of the proposed approach is compared against the state-of-the-art techniques and experiments are carried out for baseline IR model [6], word embedding based [1], and pseudo relevance feedback (PRF) based [32] query expansion, retrieval models. In the baseline IR model, query terms are directly matched and documents are retrieved based on matching scores. Tables 8 and 9 show the results obtained for the above-mentioned models using the same metrics as used for the phrase embedding model. From tables 7, 8, and 9 it is observed that the proposed phrase embedding model has outperformed both in terms of relevance and the average retrieval time. Although the Baseline IR model is faster it generates significantly less relevant results compared to the phrase embedding model. Figure 9 shows the comparison of the average time taken by each retrieval model and graphically represent it in a bar chart.

Table 8: Quality of returned documents in terms of P@5, P@10, MAP, and Rprec metrics for OHSUMED MedLine dataset (ART is an average retrieval time in milliseconds taken by model to retrieve top 100 documents)

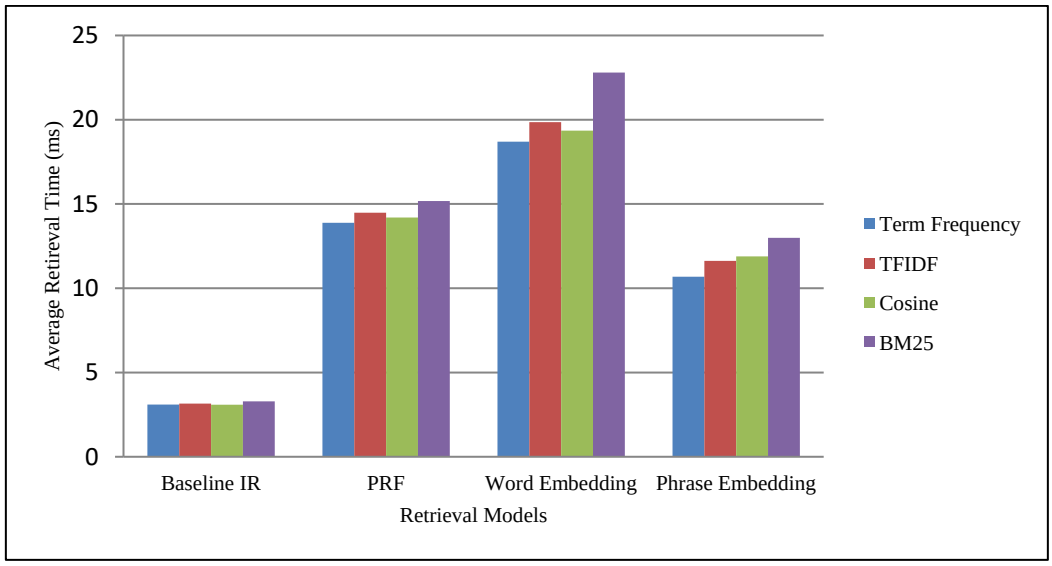
Techniques	**Scoring Function**	**P@5**	**P@10**	**MAP**	**Rprec**	**ART/Query @100 in ms**
Baseline [6]	**Term Frequency [28]**	0.3095	0.3048	0.1455	0.3161	2.5690
	TFIDF [28]	0.3175	0.3286	0.1533	0.3314	2.7377
	Cosine [29]	0.3397	0.3302	0.1612	0.3439	2.6964
	BM25 [30]	0.3714	0.3651	0.1762	0.3745	2.7541
PRF [32]	**Term Frequency [28]**	0.3937	0.4238	0.2186	0.4396	10.4888
	TFIDF [28]	0.4127	0.4254	0.2415	0.4622	11.4139
	Cosine [29]	0.4130	0.4206	0.2399	0.4470	11.1384
	BM25 [30]	0.4476	0.4508	0.2620	0.4709	11.8932
Word Embedding [1]	**Term Frequency [28]**	0.4222	0.4333	0.2503	0.4377	17.5101
	TFIDF [28]	0.4540	0.4571	0.2662	0.4766	17.9349
	Cosine [29]	0.4762	0.4778	0.2765	0.4892	17.7742
	BM25 [30]	0.4794	0.4952	0.2852	0.4925	18.4628

Table 9: Quality of returned documents in terms of P@5, P@10, MAP and Rprec metrics for TREC-CDS 2014 dataset (ART is an average retrieval time in milliseconds taken by model to retrieve top 100 documents)

Techniques	**Scoring Function**	**P@5**	**P@10**	**MAP**	**Rprec**	**ART/Query @10 in ms**
Baseline [6]	**Term Frequency [28]**	0.3400	0.3567	0.1398	0.3589	3.1011
	TFIDF [28]	0.3267	0.3599	0.1402	0.3754	3.1550
	Cosine [29]	0.3467	0.3601	0.1435	0.3623	3.0901
	BM25 [30]	0.3533	0.3600	0.1446	0.3636	3.2854
PRF [32]	**Term Frequency [28]**	0.3800	0.3567	0.1726	0.4201	13.8876
	TFIDF [28]	0.3733	0.4167	0.2069	0.4451	14.4870
	Cosine [29]	0.3735	0.4133	0.2092	0.4506	14.1960
	BM25 [30]	0.4333	0.4433	0.2166	0.4659	15.1713
Word Embedding [1]	**Term Frequency [28]**	0.4332	0.4434	0.2142	0.4638	18.6973
	TFIDF [28]	0.4668	0.4967	0.2618	0.5034	19.8605
	Cosine [29]	0.4532	0.4933	0.2534	0.4975	19.3498
	BM25 [30]	0.4533	0.4967	0.2716	0.5229	22.7994



(a) OHSUMED Dataset



(b) TREC- CDS 2014 Dataset

Fig. 9: Comparative analysis of Average Retrieval Time taken by different retrieval models using various matching techniques

5.0 CONCLUSION

This paper has proposed a spark enhanced neural network phrase embedding framework for query expansion. To learn phrase embeddings two benchmark datasets are used. In recent times, query expansion has been performed using word embeddings which shows interesting semantic interconnections amongst different terms; however, it is unable to find relationships between phrases and their contextually similar forms. More specifically, this study has implemented a chunking technique in a distributed environment, which is created using a spark map-reduce framework to extract phrases and tagged it to create the new annotated dataset. With these newly annotated datasets, phrase embeddings are learned using the word2vec CBOW model. Query expansion is performed by extracting contextually similar words or phrases with the help of the proposed phrase embedding model and its impact on retrieving relevant information is studied. Several experiments are conducted to evaluate the effectiveness of the proposed method over two standard datasets viz. - TREC-9 OHSUMED and TREC-2014 clinical decision support document collections. We also performed a comparative analysis with state-of-the-art retrieval models. It is evident from exhaustive experiments that phrase embedding technique with

BM25 similarity measure has outperformed other state-of-the-art retrieving models. In the future, we plan to investigate the use of phrase embedding in clinical trial predictions.

REFERENCES

[1] Mikolov T, Chen K, Corrado G, Dean J, "Efficient estimation of word representations in vector space", *arXiv preprint*, arXiv:1301.3781. 16 January 2013.

[2] Pennington J, Socher R, Manning C, "Glove: Global vectors for word representation", *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532-1543.

[3] Bengio Y, Ducharme R, Vincent P, Jauvin C. "A neural probabilistic language model", *Journal of machine learning research*,3 February 2003, pp. 1137-55.

[4] Collobert R, Weston J, "A unified architecture for natural language processing: Deep neural networks with multitask learning", *In Proceedings of the 25th international conference on Machine learning*, 5 July 2008, pp. 160-167.

[5] Salton G, Wong A, Yang CS, "A vector space model for automatic indexing", *Communications of the ACM*, 1 November 1975, pp. 613-20.

[6] Salton G, & McGill M, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

[7] Lund K, Burgess C, "Producing high-dimensional semantic spaces from lexical co-occurrence", *Behavior research methods, instruments, & computers*, Vol. 28 No. 1, June 1996, pp. 203-8.

[8] Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R, "Indexing by latent semantic analysis", *Journal of the American society for information science*, Vol. 41 No. 6, September 1990 , pp. 391-407.

[9] Ji S, Yun H, Yanardag P, Matsushima S, Vishwanathan SV, "Wordrank: Learning word embeddings via robust ranking". *arXiv preprint*, arXiv:1506.02761. 9 June 2015.

[10] Bojanowski P, Grave E, Joulin A, Mikolov T, "Enriching word vectors with subword information", *Transactions of the Association for Computational Linguistics*, December 2017, pp. 135-46.

[11] Le Q, Mikolov T, "Distributed representations of sentences and documents", *In International conference on machine learning*, 27 January 2014, pp. 1188-1196.

[12] Zheng G, Callan J, "Learning to reweight terms with distributed representations", *In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 9 August 2015, pp. 575-584.

[13] Nalisnick E, Mitra B, Craswell N, Caruana R, "Improving document ranking with dual word embeddings", *In Proceedings of the 25th International Conference Companion on World Wide Web*, 11 April 2016, pp. 83-84.

[14] Ganguly D, Roy D, Mitra M, Jones G J, "Word embedding based generalized language model for information retrieval", *In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 9 August 2015, pp. 795-798.

[15] Zuccon G, Koopman B, Bruza P, Azzopardi L. "Integrating and evaluating neural word embeddings in information retrieval", *In Proceedings of the 20th Australasian document computing symposium*, 8 December 2015, pp. 12.

[16] Vuli I, Moens M F, "Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings", *In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*,9 August 2015, pp. 363-372.

[17] Socher R, Manning C D, Ng A Y, "Learning continuous phrase representations and syntactic parsing with recursive neural networks", *In Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop"*, Vol. 2010, 10 December 2010, pp. 1-9.

[18] Blacoe W, Lapata M, "A comparison of vector-based representations for semantic composition", *In Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, 12 July 2012, pp. 546-556.

[19] Yin W, Schtze H, "An exploration of embeddings for generalized phrases", *In Proceedings of the ACL 2014 Student Research Workshop*, 2014, pp. 41-47.

[20] Yu M, Dredze M, "Learning composition models for phrase embeddings", *Transactions of the Association for Computational Linguistics*, December 2015, pp. 227-42.

[21] Li M, Lu Q, Xiong D, Long Y, "Phrase embedding learning based on external and internal context with compositionality constraint", *Knowledge-Based Systems*, 15 July 2018, pp. 107-16.

[22] Kadir, Rabiah A., Rufai Aliyu Yauri, and Azreen Azman, "Semantic ambiguous query formulation using statistical Linguistics technique", *Malaysian Journal of Computer Science*, 28 December 2018, pp. 48-56.

[23] Roy D, Paul D, Mitra M, Garain U, "Using word embeddings for automatic query expansion", *arXiv preprint*, arXiv:1606.07608, 24 June 2016.

[24] Rekabsaz N, Lupu M, Hanbury A, "Uncertainty in neural network word embedding: Exploration of threshold for similarity", *arXiv preprint*, arXiv:1606.06086. 20 June 2016.

[25] ALMasri M, Berrut C, Chevallet JP, "A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information", *In European conference on information retrieval*, *Springer, Cham*, March 2016, pp. 709-715.

[26] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J, "Distributed representations of words and phrases and their compositionality", *In Advances in neural information processing systems*, 2013, pp. 3111-3119.

[27] Abney SP, "Parsing by chunks, In Principle-based parsing", *Springer, Dordrecht*, 1991, pp. 257-278.

[28] Salton, Gerard, and Christopher Buckley, "Term-weighting approaches in automatic text retrieval." *Information processing & management*, Vol. 24 No. 5, 1988, pp. 513-523.

[29] Singhal A, "Modern information retrieval: A brief overview", *IEEE Data Eng. Bul.*, Vol. 24 No. 4, 2001, pp. 35-43.

[30] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M., "Okapi at TREC-3", *Nist Special Publication*, 1995, pp. 109.

[31] Thakur N, Mehrotra D, Bansal A, Bala M, "Comparative analysis of ranking functions for retrieving information from medical repository", *Malaysian Journal of Computer Science*, Vol. 32 No. 1, 31 January 2019, pp. 18-30.

[32] Buckley, C., Salton, G., Allan, J., & Singhal, A, "Automatic query expansion using SMART: TREC 3", *NIST special publication*, 1995, pp. 69-69.