Improving Ranking Using Hybrid Custom Embedding Models on Persian Web

Shekoofe Bostan, Ali Mohammad Zareh Bidoki*
and Mohammad-Reza Pajoohan

Department of Computer Engineering, Yazd University, Iran
E-mail: sbostan@stu.yazd.ac.ir; alizareh@yazd.ac.ir; pajoohan@yazd.ac.ir
**Corresponding Author*

Received 24 August 2023; Accepted 23 October 2023;
Publication 19 December 2023

Abstract

Ranking plays a crucial role in information retrieval systems, especially in the context of web search engines. This article presents a new ranking approach that utilizes semantic vectors and embedding models to enhance the accuracy of web document ranking, particularly in languages with complex structures like Persian. The article utilizes two real-world datasets, one obtained through web crawling to collect a large-scale Persian web corpus, and the other consisting of real user queries and web documents labeled with a relevancy score. The datasets are used to train embedding models using a combination of static Word2Vec and dynamic BERT algorithms. The proposed hybrid ranking formula incorporates these semantic vectors and presents a novel approach to document ranking called HybridMaxSim. Experiments conducted indicate that the HybridMaxSim formula is effective in enhancing the precision of web document ranking up to 0.87 according to the nDCG criterion.

Keywords: Word embedding, Word2Vec, BERT, semantic vector, query, ranking.

Journal of Web Engineering, Vol. 22_5, 797–820.
doi: 10.13052/jwe1540-9589.2253

1 Introduction

Web information retrieval is the process of searching within web document collections for a special query. Semantic search is a concept to improve the accuracy of the search process by understanding the searcher's intent and decreasing ambiguity. Natural language processing (NLP) techniques are useful for semantic search [1]. These techniques make it possible to understand the context of the content. The definition of contextual retrieval is dependent on surrounding words in text content [2]. Several studies have been realized for contextual text meaning based on different languages. However, few works are interested in the Persian language. Ranking web documents is an important aspect of information retrieval and search engines. It involves sorting and ordering documents based on certain criteria to provide users with accurate and relevant results. Understanding the real meaning of the query can be used to prioritize relevant results. A word with different meanings may lead to deviation of the query from the searcher's intent. Recently, researchers have shown an increased interest in word embedding techniques. Word embedding is a technique used in natural language processing to represent words as real-valued vectors that capture semantic relationships and syntactic similarity. This distributed representation for text is obtained using language modeling and feature learning techniques, with different approaches including vectors of co-occurring words or linguistic contexts. Word embedding has various applications in NLP, enabling computers to understand and process text-based content more effectively. They provide a powerful way to represent words numerically, revolutionizing the field of NLP and contributing to significant advancements in language processing tasks [3].

The aim of this research is to improve search result rankings by understanding the meaning of Persian phrases through semantic embedding. Different embedding algorithms are investigated, and ultimately Word2Vec and BERT embeddings are used to train the models on web documents. Instead of expanding queries using term embedding, the study focuses on converting queries and documents into embedding vectors and ranking them based on vector similarities using a proposed hybrid ranking formula. The paper is organized as follows. In the next section, a comprehensive study of previous works on word representation and ranking mechanisms are reviewed. In Section 3, an outline of the methodology used to train the models with Word2Vec and BERT embeddings is explained. In Section 4, a proposed ranking approach will be described. Section 5 specifies the experimental analysis and, finally, the paper is concluded in Section 6.

2 Previous Work

Previous research has been categorized into three distinct categories. The first category pertains to word embedding approaches, while the second category focuses on related work on Persian word embedding. Lastly, the third category delves into ranking approaches.

2.1 Word Embedding Approaches

Natural language processing is a subfield of machine learning that is often used for text processing. The text consists of smaller parts, such as words and characters. The numerical representation of words and text is a prerequisite for most machine learning algorithms. Our goal is to use semantic vectors in information retrieval and ranking. An important purpose is to achieve a meaningful representation of the query to lead to a deep understanding of its meaning [4]. Traditional vector methods such as BoW [5], which stands for Bag of Words, and TF-IDF [6], which stands for term frequency times the inverse of the document frequency, convert text sentences into numeric vectors. These models use word frequency but the grammar, meaning, order, and conceptual connection between words are ignored. In 2003, Benjiou et al. introduced a model consisting of a neural network with one hidden layer to predict the next word in the text [7]. This concept was introduced as word embedding which tried to represent a word in n-dimensional space. In 2013, Google's Word2Vec algorithm [8] was introduced by Miklow et al. It contains two architectures known as the Continuous Bag of Word (CBOW) model to predict the current word based on the context, and the Skip-gram model to predict surrounding words of a given current word.

The global vector model, GloVe [9], was proposed in 2014 by Pennington et al. at Stanford University. The GloVe model focuses on word co-occurrences over the whole corpus. In 2014, the FastText algorithm was introduced by Facebook which is very similar to the Word2Vec idea. In this model, instead of using the whole word, part of the word and characters are considered. Output embedding of each word is a combination of lower-level embeddings of sub-words and characters.

Contextual embedding assigns each word a representation based on its context. Word2Vec uses only one weighted layer called word embedding but a neural network can contain many layers that increase the power and complexity of the network. One of the limitations of Word2Vec is learning a fixed embedding for each vocabulary word. The ELMo method was introduced in 2018 as a new type of deep contextualized word representation

model with a deep understanding of the syntax and semantics of words [11]. Unlike traditional word embedding, the vector of representing a word in two sentences is different. ELMo uses next-word prediction in a sequence of words based on language model. It is based on LSTM [12] architecture which is a type of RNN (recurrent neural network) [13]. ELMo uses a bidirectional language model that combines two LSTMs in forward and backward directions to give a sequence of tokens and predict the probability of the next word.

In 2018, Devlin et al. introduce a new language representation model called BERT [14] which is an acronym for bidirectional encoder representations from transformers. BERT is a pre-train deep bidirectional representation of unlabeled text in multiple layers and can be fine-tuned with one additional output layer to create models for important tasks. BERT is based on transformers and is logically similar to ELMo. In fact, BERT is a combination of bidirectional RNN and deep RNN architecture known as bidirectional deep RNN. It consists of a pre-training step on unlabeled data and fine-tuning step using labeled data from the downstream tasks [14]. ALBERT [15], RoBERTa [16] and DistilBERT [17] were proposed as expanded algorithms of the BERT algorithm in 2019.

OpenAI released GPT-3 in 2020 which is an acronym for the generative pre-trained transformer in continuation of GPT-2 and GPT with increasing the number of parameters and training on larger data. Each layer consists of two sublayers that include multi-head self-attention mechanism and a fully connected network [18]. GPT-4 was released in 2023 which is the latest milestone in OpenAI's effort in scaling up deep learning [19].

2.2 Related Work on Persian Word Embedding

Despite considerable research interest in word embedding in recent years, few works have studied these models in the Persian language. The Polyglot project [19] trained word embedding for more than 100 languages including Persian using Wikipedia articles. Facebook published pre-trained fastText vectors for many languages on Wikipedia [20]. In 2018 the impact of the corpus domain on Persian word representation had been researched [21]. In 2018, Zahedi et al. introduced a comparison between static models that indicate the outperform of FastText and Word2Vec [22]. In 2020 in the dynamic embeddings, ParsBERT model was introduced based on BERT architecture [23]. In 2023, Bostan et al. introduced Persian BERT model and evaluated with ParsBERT and BERT multilingual language models [24].

2.3 Ranking Approach

Ranking documents involves sorting them based on their relevance to a query. Document ranking based on embedding vectors is a technique used to rank and retrieve documents based on their similarity to a query. Embedding vectors represent documents and queries in a high-dimensional space, where the similarity between vectors indicates the relevance of the document to the query. In 2016, the generative topic modeling was introduced, which is a combination of word embedding and topic modeling. This model represents documents as fixed-length feature vectors in a low-dimensional continuous space under the topics [25]. In 2016, the DESM model was proposed as a dual embedding model for document ranking based on the Word2Vec algorithm, which focuses on training words in documents and queries [26]. In 2017, Dehghani et al. presented a weakly supervised neural model. In this approach, the output of an unsupervised ranking model, such as BM25, was used as a weak supervision signal [27]. K-NRM is a kernel-based neural model for document ranking that utilizes a translation matrix to model word-level similarities through word embeddings, which was introduced in 2017 [28]. In 2019, a matrix factorization approach was proposed for node embedding in a network of documents. This approach was inspired by the GloVe algorithm, which is based on the co-occurrence probability of words [29]. In 2020, a method called Gaussian embedding of linked documents was introduced, which focuses on embedding linked documents into a pre-trained semantic space consisting of a set of pre-trained embedding vectors [30]. A novel approach aiming to improve document ranking based on semantic similarity measurement and the factor of association was proposed in 2020. Semantic similarity focuses on retrieving similar textual documents based on a focused query, while the association factor emphasizes on constructing a kernel-based neural model [31].

In 2022, entity embedding based on relations for better document ranking was examined, utilizing a neural network for embedding Wikipedia documents based on graphs [32]. A machine learning algorithm based on re-ranking documents was introduced in 2022. The ranking structure involves encoding queries and documents using the BERT algorithm, and then using a re-ranking learning model based on TFR to improve results and optimize ranking performance [33]. Additionally, in 2022, some pre-trained embedding methods were evaluated to identify the best model for document ranking. The evaluation results indicate better performance of the joint sentence encoder and SentenceBERT [34].

3 Custom Embedding Model

The approach of utilizing embedding in information retrieval (IR) entails creating a compact vector representation for both queries and documents within the text. This involves transforming queries and documents into embedding vectors within an n-dimensional space. The ranking process is determined by the similarity ranking method of these vectors, which will be further elaborated on in the subsequent section.

Figure 1 shows that this article uses two embedding algorithms to improve document ranking. The first algorithm, Word2Vec, embeds static vocabulary and is trained on Persian web documents. The second algorithm, BERT, embeds sentences and uses the pre-trained ParsBERT model for training, with a custom fine-tuning process. This article presents a combined approach of the two models for better document ranking, with detailed explanations for each part.

3.1 Embedding with Persian Word2Vec Model

Word2Vec is a popular embedding model in natural language processing that represents words or phrases as dense vectors in a continuous vector space. It is designed to capture the semantic and syntactic similarities between words. Developed by Tomas Mikolov et al. at Google, Word2Vec has gained significant attention for its ability to learn meaningful word representations from large amounts of text data. The basic idea behind Word2Vec is that words with similar meanings tend to appear in similar contexts. Therefore, the model learns word embeddings by training on large textual datasets, predicting the context or neighboring words of a target word. Word2Vec offers two primary architectures for learning these embeddings: Continuous Bag of Words (CBOW) and skip-gram. In the CBOW architecture, the model tries to predict the target word based on its surrounding context words. The context words are used as input to the model, and the goal is to predict the target word. The CBOW model learns by updating the word embeddings in a way that the predicted target word maximizes the likelihood of the observed context words. This architecture works well when the target word is influenced by the neighboring words in its context. The skip-gram architecture, on the other hand, is the reverse of CBOW. It predicts the context words given a target word. The skip-gram model takes a target word as input and generates a probability distribution over the context words. The parameters of the model, i.e., the word embeddings, are updated to maximize the likelihood of the observed context words. Skip-gram is known to perform better when applied

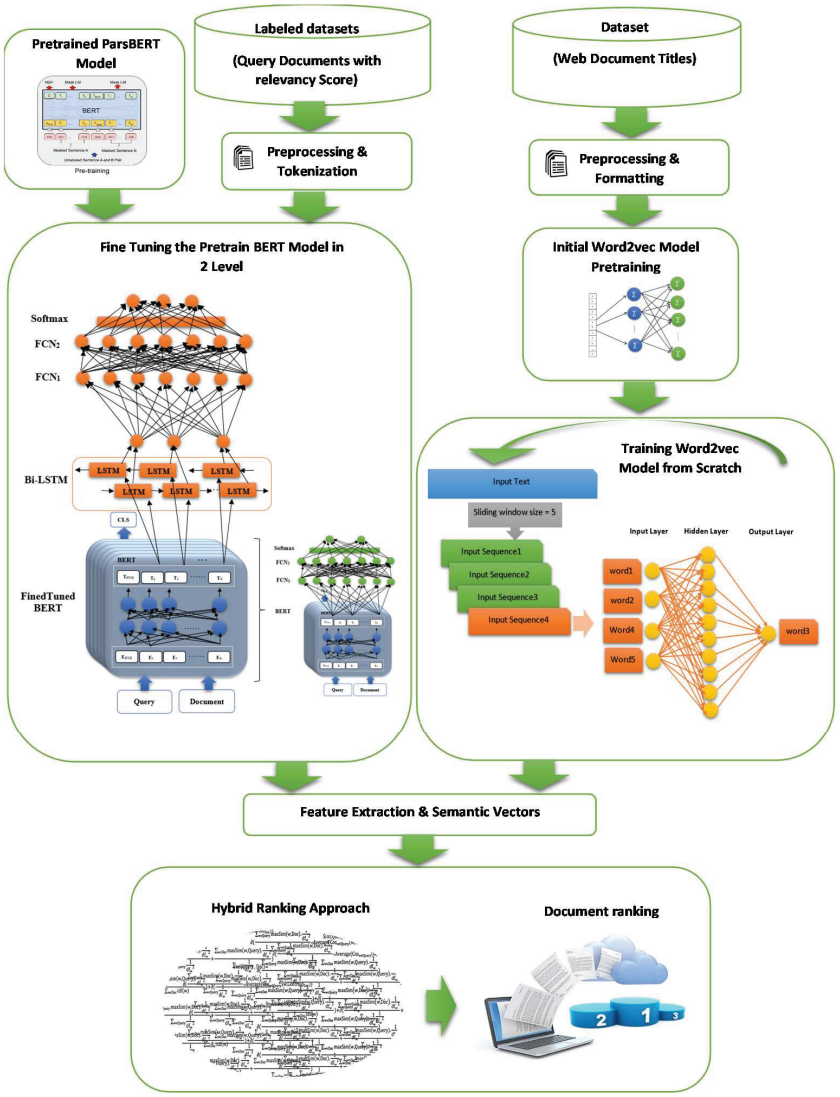

Figure 1 The proposed framework for hybrid model ranking.

to large datasets and is useful for capturing less frequent words and their relationships. The output of training Word2Vec is a set of word vectors, where each word is represented by a high-dimensional vector, typically with hundreds of dimensions. These vectors capture semantic relationships between words, with similar words having close vector representations.

3.1.1 Data gathering

In order to train word embedding vectors, a substantial corpus of sentences is required. To enhance the accuracy and usefulness of our pre-trained model at word, phrase, and sentence levels, we have utilized a novel form of corpus. The website title serves as a concise and precise preview of the content covered on web pages. Hence, we have employed web page titles as part of our training process. The corpus used in our study has been carefully extracted and collected by the Parsijoo search engine, comprising an extensive collection of 681 million titles.

One important aspect of working with Persian text is the normalization process. This is because Persian text often contains Arabic characters and numbers, which need to be converted to their Persian equivalents. In contrast, English text is typically already in a standardized format and does not require this type of normalization. Spacing between words is another crucial aspect to consider in Persian text. For instance, the name of a famous Iranian actor, Mohammad Reza Golzar, can be written as one word or three separate words with spaces in between. This distinction is more significant in Persian text than in English and can impact text analysis and ranking. In addition, Persian text tends to be longer and more complex than English text, making it more difficult to understand and analyze. This means that we need to take extra care when processing and ranking Persian text, and we may need to use more sophisticated algorithms to accurately capture the meaning of the text. Lastly, the position of words in Persian text is crucial since Persian is a highly inflected language. The position of words in a sentence can affect their meaning, which is another factor to consider when ranking Persian text. Persian relies heavily on context to convey meaning. Understanding the context in which a sentence is used can help in understanding the intended meaning of the text. We applied several steps to preprocess the corpus to transform it into a proper format. The first step involves normalizing the characters and handling invalid characters by either removing them or converting them to standard characters. Additionally, Arabic characters are converted into Persian to ensure consistency. After the initial preprocessing, further steps are taken that include removing punctuations and numbers from the text. Moreover, Persian tokens with less than three characters are disregarded to maintain focus on meaningful content. It is important to note that the corpus comprises texts in languages other than Persian. Consequently, during the model training process, non-Persian words are incorporated as well, allowing the model to learn from this diverse linguistic data.

3.1.2 Methodology

This section focuses on the implementation and training of the Continuous Bag-of-Words (CBOW) model of the Word2Vec algorithm. We leverage a large collection of Persian web documents for training the model. The implementation of the CBOW model for Persian web documents comprises several steps. The preprocessed documents are tokenized into meaningful words to form a corpus for training the CBOW model. We leverage the Python programming language and appropriate libraries, such as Gensim, for the implementation and training of the CBOW model. This involved converting words into 100 dimensional vectors to capture their contextual meaning. We set a minimum word frequency of 20, indicating that any words appearing less frequently were disregarded for the training process. To capture local word dependencies, we utilized sliding windows of length 5.

3.1.3 Training process

The training process involves feeding the CBOW model with input data consisting of word-context pairs extracted from the preprocessed Persian web documents. The model is trained using a neural network with a hidden layer, optimizing the parameters to learn meaningful word embeddings. This allows for capturing the semantic relationships between words, enabling applications such as similarity measurement and information retrieval within the Persian language domain. Our model is trained using the Continuous Bag of Words (CBOW) structure, where the aim is to predict the target word based on its surrounding context words. We initially pretrained the model using a subset of the corpus, gradually adding newer data in subsequent training steps for continued improvement. In the paper, a set of web documents is provided as the basis for building an embedding model and generating semantic word vectors. The collection of documents, comprises a set of individual documents. The vocabulary consists of unique terms extracted from the documents. The next step involves extracting word representations for all the words in the vocabulary using the Word2Vec model. This process allows for the creation of numerical vectors that capture the semantic meanings of the words, enabling further analysis and exploration of the document collection in proposed ranking formulas. The process starts by inputting a set of sentences into the model. Then, the training process commences, during which the model utilizes a dictionary of trained words. In each iteration, the weights of the words are updated. Once the training is complete, the model is saved, allowing it to be used for further training with a new set of documents

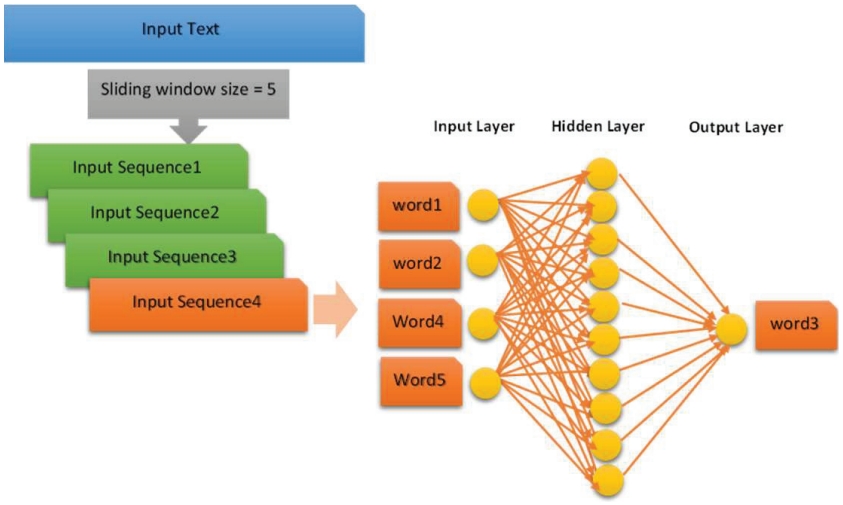

Figure 2 An example of training the CBOW model.

in the future. This feature enables the model to be retrained using a more recent corpus. During each step of the retraining process, the previous model and the last dictionary of vocabulary are loaded, ensuring that the training continues seamlessly. In Figure 2, the training process based on the CBOW model is depicted.

The training begins by traversing the training corpus using a sliding window technique, which selects input sequences from the text. Within each input sentence, the middle word is masked, creating a scenario where the model needs to predict the missing word. This prediction task is performed by passing the masked sentence through a fully connected dense layer, which serves as a hidden layer in the CBOW model. The dense layer utilizes the current weights of words and their corresponding features to generate a prediction for the missing word. This iterative process continues, updating the weights of words and their features, until all sentences have been trained through the sliding window technique.

In this research, a significant volume of web document titles, totaling 54 million, was incorporated into the model at each step. The training process for each document category spanned approximately 8 hours, resulting in a total training duration of 100 hours for the extensive dataset of 681 million documents. Throughout the training, a vast collection of approximately 2 million unique words were obtained. However, to optimize the usefulness of the vocabulary, words that occurred less than 20 times were eliminated. As a

Table 1 Training information

Subject	Values
Train time	100 hours
Title counts	681 million
Size	75 GB
Word embedding model	Word2Vec (CBOW)
Trained words count	329,836
Sliding window size	5
Vector dimension size	100
Minimum word frequency	20
Number of iterations	3

result, the number of trained words was subsequently reduced to 329,836, as summarized in Table 1.

3.2 Embedding with the Persian BERT Model

This section focuses on the utilization of the BERT algorithm for processing Persian web documents. BERT is a transformer-based model that is pre-trained on a large corpus of text data to learn contextualized word representations. Unlike traditional NLP models that process words in a sequential manner, BERT incorporates bidirectional training, allowing it to consider the full context of a word by looking at both the preceding and following words. During pre-training, BERT is trained on a large amount of text data, often using unsupervised techniques. The model learns to predict missing words in a sentence based on the surrounding context, task known as masked language modeling. Additionally, BERT performs another pre-training task called next sentence prediction, where it learns to predict whether two sentences appear consecutively in the original text. BERT embeddings are typically obtained by fine-tuning the pre-trained BERT model on specific downstream tasks by adding task-specific layers or classifiers on top of the pre-trained model.

3.2.1 Methodology

In our article, we utilize the pre-trained Pars-BERT [23] embedding model and fine-tune it for our purpose. The Pars-BERT embedding model is a pre-trained language model that has been specifically designed for the Persian language. It utilizes the bidirectional encoder representations from transformers (BERT) architecture and has been trained on a large corpus of Persian text. By adding multiple layers on top of the pre-trained model, we achieve a better performance and accuracy.

3.2.2 Data gathering

The BERT model needs labeled data for fine-tuning, which includes pairs of queries and documents collected from various websites and labeled by a team of experts. The first dataset has 200 queries with an average of 15 labeled documents per query, while the second dataset has 3000 queries with three labeled documents per query. For the first dataset, the label zero indicates that the document is irrelevant to the query, while label one indicates that it is relevant. To increase the accuracy of the model during training for the second dataset, three types of labels based on the degree of relevance between the query and document are used, entailment, contradiction, and neutral. The entailment label is assigned to a document that is completely relevant to the query in terms of meaning. By contrast, the contradiction label is assigned to a document that is visually similar to the query but completely different in meaning. For example, a document with the content "the latest Bugatti car model" is considered as a contradiction with the query "the latest Cerato car model". The neutral label is assigned to a document that has a general content without any specific direction. For example, a document with the content "news about cars around the world" is an example of this label. These datasets are used for the first and second fine-tuning processes, respectively. The datasets are first preprocessed and normalized. In this process, 70% of the data is used for training, and 30% of it is randomly divided into two 15% sets for evaluation and testing purposes.

3.2.3 Fine-tuning

Fine-tuning a pre-trained model requires much less cost compared to training a model from scratch. By fine-tuning the model towards document classification based on the user's query, the model learns to determine the similarity between two expressions in the form of a query and a document. In the beginning, the pre-trained Pars-BERT model is loaded, and then tokenization is performed for the training, evaluation, and testing sets. Since the labels are based on determining the similarity between documents and queries, a custom data access and processing method is created. Then, the dataset is divided into smaller categories and, after injection into the network, it is shuffled. In each training epoch, the weights are updated using fixed parameters and added layers on top of the model. The average cost is calculated during the training process. The model architecture consists of two fully connected layers for weighting the hidden layer and using the SoftMax for the output layer. This model returns two outputs, one for the language mask model and the other for predicting the next sentence. The output of predicting the next sentence

in the first layer of the model is passed to the fully connected network. Then, by passing through the cost function, 1% of the weight of the hidden layer is randomly forgotten to make it more regularized and reduce the testing error. This output is then passed to the second layer of the fully connected network and then through the softMax. The final output of the model indicates whether the second sentence is related to the first or not. With a given number of expected class labels, the model training begins. During this process, some of the pre-trained weights remain unused, while others are randomly initialized. The pre-trained model head layer is removed, and a classification head with randomly initialized values is used instead. During training, this new head is fine-tuned on the sequence of classification tasks, and the pre-trained model knowledge is transferred to it. The training process continues for three epochs, and finally, the resulting model with new weights is saved for future use.

In the second fine-tuning stage, a different structure with a separate dataset and format is used. The fine-tuned model from the previous stage is used as input for this stage. In fact, the output of the first fine-tuning stage is used as input for the second fine-tuning stage to improve the weights for both structures, which can be seen in Figure 3.

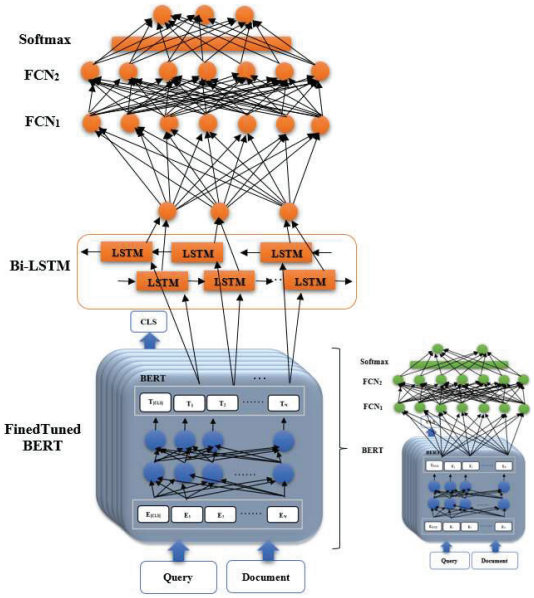

Figure 3 Custom fine-tuning.

In this structure, a bidirectional LSTM layer is added at the top of the model. The input sequence is passed through the LSTM from both directions simultaneously. After concatenating the outputs, 3% of the weight of the hidden layer is randomly forgotten. Then it passes through the fully connected network layers and are activated by the softMax function. At this stage, training is only applied to the top layers to extract features, but it is also possible to use the pre-trained model. After extracting the features of the pre-trained model, the fine-tuning process is performed based on the desired task to cover new data. Therefore, the model is first removed from the frozen state.

4 Ranking

The purpose of this research is to enhance the ranking of documents by leveraging semantic vectors, aiming for higher accuracy while minimizing the associated costs. To achieve this goal, the study focuses on calculating the similarity between semantic query and document vectors. The article introduces a novel approach that involves extracting semantic vectors of words and texts and incorporating them into hybrid ranking formulas. In this approach, the semantic vectors are represented in a multidimensional space and ranked accordingly within that space. The ranking process is based on calculating the cosine of the angle between two vectors, but with the utilization of a newly proposed structure and relationships specific to the multidimensional space. Through this innovative methodology, the research aims to achieve improved document ranking, combining the power of semantic vectors with the advancements in multidimensional ranking techniques.

This paper focuses on utilizing embedding vectors to improve the ranking of query phrases and documents. We employed two powerful models, the Persian Word2Vec model and the fine-tuned BERT model, both of which were trained, in Section 3, to extract high-quality embedding vectors. We propose a novel combined formula for enhanced document ranking. By leveraging the power of embedding vectors and incorporating a ranking formula, this paper aims to enhance the precision and effectiveness of document retrieval and ranking systems. According to Figure 4, a set of queries and labelled web title documents is needed. ALL queries and documents are collected from various sites such as Digikala and Aparat. This dataset contains 100 queries, for each query, an average of 10 labelled documents have been prepared. Labels from 0 to 5 indicate the relevance of the document to the query. Larger number means higher relevancy.

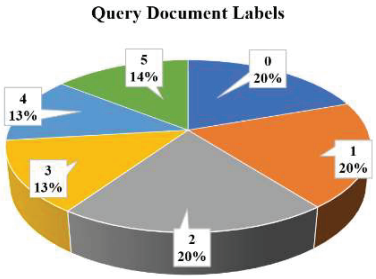

Figure 4 Label details of evaluated documents.

4.1 SentenceModel Ranking Formula

Cosine similarity [20] is a measure used to determine the similarity between two vectors in a multi-dimensional space. It calculates the cosine of the angle between the vectors, indicating their directional similarity rather than their magnitude. The cosine similarity ranges from -1 to 1, where a value approaching 1 signifies high similarity, while a value approaching -1 indicates dissimilarity.

Each query q and document d, consists of a set of different tokens can be presented as,

$$q = \{t_1, t_2, \ldots, t_k\}, \quad d = \{t_1, t_2, \ldots, t_m\} \tag{1}$$

Ranking documents based on embedding vectors of query and document is called SentenceModel similarity by calculating the average of vectors as follows,

$$\text{Similarity} = \cos(\theta) = \frac{\vec{e}_q \cdot \vec{e}_d}{|\vec{e}_q||\vec{e}_d|} \tag{2}$$

4.2 MaxSim Ranking Formula

An alternative approach involves utilizing the semantic vectors of individual words within each sentence and then evaluating the degree of similarity between them. This approach is employed in the maxSim [26] ranking formula, which uses two text segments to define semantic similarity. The segments used in this study are the query and document contents. First for each word w in query q, try to identify the word in document d with the highest semantic similarity, using the cos similarity of vectors. Next, the same process for each w in d is applied to determine the most similar words. The importance of a word is determined using the inverse document

frequency (IDF). The value of ∂ is 0.5. The similarity between q and d is determined using scoring function, $sim_{maxSim}(q.d)$.

$$sim_{maxSim}(q.d) = \partial \left(\frac{\sum_{w \in \{q\}} (maxSim(w.d) * idf(w))}{\sum_{w \in \{q\}} idf(w)} + \frac{\sum_{w \in \{d\}} (maxSim(w.q) * idf(w))}{\sum_{w \in \{d\}} idf(w)} \right) \tag{3}$$

The similarity is calculated for each pair of query and document separately. Then the documents are sorted and ranked based on obtained similarities.

4.3 Proposed Hybrid Ranking Formula

Using a hybrid ranking model that combines various models and approaches provides several benefits. By combining different models and approaches, a hybrid ranking model leverages the strengths of each component to achieve better performance. Each model excels in different aspects of ranking, and combining them leads to more accurate and robust results. The hybrid model has a more comprehensive and diverse representation of the data, leading to a better understanding of the ranking task [37].

This paper proposes a hybrid ranking formula that combines Word2Vec and BERT models to capture both semantic and contextual information, resulting in more accurate rankings. This hybrid model is more robust to noise, outliers, or biases present in the data and can handle different types of data by utilizing specialized models for each data type [38]. The hybrid model is particularly useful for document ranking, as it leverages the strengths of both models to produce a comprehensive and diverse representation of the data [39].

In this stage, a trained Word2Vec model and fine-tuned BERT model is used to calculate the maximum similarity between the word vector of the first text and the entire sentence of the second text. This approach is different from calculating the maximum similarity between individual words from both texts.

The embedding vector of each token as t is extracted from the trained Word2Vec model as follows,

$$\vec{e}_{t_i} = \mathrm{M}_{\mathrm{Word2Vec}}(t_i) \tag{4}$$

Also, the embedding vectors of the query and document sentence as s_i are extracted from the fine-tuned BERT model as follows:

$$\vec{e}_{s_i} = \mathrm{M}_{\mathrm{BERT}}(s_i), \text{ where } q \text{ and } d \in S \tag{5}$$

This new approach is called HybridMaxSim which is represented as follows.

$$\begin{aligned} &sim_{HybridMaxSim}(q.d) \\ &= \partial \left(\frac{\sum_{w \in \{q\}} (maxSim(\vec{w}_{word2vec} \cdot \vec{d}_{BERT}) * idf(w))}{\sum_{w \in \{q\}} idf(w)} \right. \\ &\left. + \frac{\sum_{w \in \{d\}} (maxSim(\vec{w}_{word2vec} \cdot \vec{q}_{BERT}) * idf(w))}{\sum_{w \in \{d\}} idf(w)} \right) \end{aligned} \tag{6}$$

The proposed method uses Word2Vec to extract semantic vectors for words in one text and a fine-tuned BERT model to extract semantic vectors for sentences in another text. By calculating the similarity between the two vectors, the keyword can be identified as the word closest to the sentence vector, indicating its relevance to the topic.

The nDCG [27] criterion is used to evaluate and compare the rating quality. nDCG is popular method for measuring the quality of search results. Cumulative gain is the sum of all the relevance scores in a sequence of retrieval order. Discounted cumulative gain (DCG) discounts the relevance score by dividing it with the log of the corresponding position. Finally, nDCG known as normalized discounted cumulative gain is the ratio of the DCG of the recommended order to the DCG of the ideal order [27]. It considers the position and the relevance score of the document in the ranked list. In this relation, which is used for the first n results, r_j represents the degree of relevance of the jth document with the corresponding query as follows,

$$\mathrm{nDCG@n} = \sum_{j=1}^{n} \frac{2^{r_j} - 1}{\log(1 + j)} \tag{7}$$

The nDCG takes into account two crucial factors: the relevance of a document to the query and its position in the ranking. To determine nDCG, relevance scores are assigned to each document in the dataset, which can be binary or ordinal, reflecting how well the document matches the user's query. The DCG is then calculated for each ranking by summing the relevance scores of the top-ranked documents, with each score discounted by its

position in the ranking. The nDCG is obtained by dividing the DCG of the ranking by the ideal DCG, which represents the maximum achievable DCG for the dataset. To calculate the ideal DCG, the documents in the dataset are sorted by their relevance scores, and the DCG for this perfect ranking is calculated. nDCG is utilized to measure how well the document ranking algorithm performs compared to an ideal ranking based on relevance scores. A higher nDCG score indicates a more precise ranking.

5 Experimental Results

The paper introduces a new hybrid ranking formula that combines Word2Vec and BERT models to improve the accuracy of rankings. The method uses semantic vectors to identify the relevance of keywords to the topic. The ranking is based on the similarity calculation between the embedding vectors in a high-dimensional space, providing a comprehensive and diverse representation of the data. The proposed approach improves traditional ranking methods by avoiding the need to expand queries and documents.

The sentenceModel similarity described in Section 4.1 serves as the baseline for this evaluation. In sentenceModel similarity, a technique for measuring sentence similarity, the cosine angle between the semantic vectors of the query and document is calculated. An approach to extract semantic vectors of the query and document sentence is by utilizing the Word2Vec model, which deals with word embedding, and calculating the pairwise average elements of the vectors to obtain a unit vector representing the meaning vector of the sentence. Alternatively, the BERT model produces a semantic vector of the input sentence. Therefore, in the first evaluation, ranking is performed based on semantic vectors of each word and calculating their average using the Word2Vec model called SentenceW2V. In the second evaluation, the same process is performed for the BERT model called SentenceBERT. The results of these two models can be seen in Table 2 which indicates an improvement in accuracy of the fine-tuned BERT model compared to trained Word2Vec model. The results indicate that BERT outperforms Word2Vec in terms of accuracy, with a score of 0.82 compared to 0.75. Another approach for ranking involves the utilization of the MaxSim formula. This approach takes into account a broader range of factors, ultimately resulting in a more comprehensive and refined ranking system. In MaxSimW2V, the trained Word2Vec model is used and the word vectors of the query and document are extracted from it. In MaxSimBERT, document ranking is based on the fine-tuned BERT model. In this method, it is necessary to extract the semantic vectors of each

Table 2 Ranking nDCG results

Ranking Formula	Model	nDCG
SentenceW2V	Trained Word2Vec	0.75
MaxSimW2V	Trained Word2Vec	0.77
SentenceBERT	Fine-tuned BERT	0.82
MaxSimBERT	Fine-tuned BERT	0.85
HybridMaxSim	Trained Word2Vec + Fine-tuned BERT	0.87

word separately. The hidden layer semantic vectors are used for this purpose. The MaxSim ranking approach yielded accuracy rankings of 0.77 and 0.85 for Word2Vec and BERT, respectively. The results of these evaluations are visible in Table 2.

The innovation of this article lies in the introduction of a novel relationship called HybridMaxSim. To take advantage of both static embedding models like Word2Vec and powerful dynamic models like BERT, a hybrid approach is introduced. This study proposes the HybridMaxSim ranking formula, which employs a fusion of trained Word2Vec and fine-tuned BERT models to achieve the highest degree of accuracy in ranking. This approach involves extracting the semantic vector of the query and document phrase from the trained Word2Vec model. Instead of computing the maximum similarity of each word in the first sentence with all the words in the second sentence, the similarity between each word in the first sentence and the embedded vector of the second sentence, based on the fine-tuned BERT model, is utilized. The HybridMaxSim ranking formula is based on a combined utilization of Word2Vec and BERT models, resulting in an accuracy of 0.87. The findings of this investigation indicate that the proposed approach yields superior accuracy compared to other techniques, as evidenced by the results presented in Table 2.

The HybridMaxSim approach combines the strengths of both models and achieves an accuracy improvement of up to 0.87. The evaluation results show that BERT is more accurate than Word2Vec, but a hybrid approach that combines the strengths of both models achieves better results.

6 Conclusion

This study aims to enhance document ranking by utilizing custom embedding models using proposed ranking approach. The results indicate that both trained Word2Vec and fine-tuned BERT models significantly improve the

accuracy and effectiveness of ranking. Although fine-tuned BERT outperformed trained Word2Vec, a hybrid approach that combined both models yielded even better results.

The proposed ranking formula integrated static Word2Vec and dynamic BERT algorithms, which were trained on two real-world datasets obtained via web crawling and user queries. The hybrid approach increased the nDCG score of web document ranking by up to 0.87. This technique has the potential to significantly enhance search engine performance and improve user satisfaction. Further research can refine the formula and explore its application in other areas of information retrieval.

References

[1] D. Lobiyal and v. Mala, "Semantic and Keyword Based Web Techniques in Information Retrieval," *International Conference on Computing, Communication and Automation (ICCCA), IEEE*, pp. 23–26, 2016.

[2] C. Mangold, "A survey and classification of semantic search approaches," *International Journal of Metadata, Semantics and Ontologies*, no. 2, pp. 23–34, 2007.

[3] F. Almeida and G. Xexeo, "Word embeddings: A survey," *arXiv preprint arXiv*, no. 1901, 2019.

[4] M. Hu, "Research on Semantic Information Retrieval Based on Improved Fish Swarm Algorithm," *Journal of Web Engineering*, no. 3, pp. 845–860, 2022.

[5] Y. Zhang, R. Jin and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, no. 1, pp. 43–52, 2010.

[6] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, no. 24, pp. 513–523, 1988.

[7] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, "A Neural Probabilistic Language Model," *Machine Learning Research*, no. 3, pp. 1137–1155, 2003.

[8] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv*, pp. 1301.3781, 2013.

[9] J. Pennington, R. Socher and C. Manning, "GloVe: Global Vectors for Word Representation," *EMNLP*, pp. 1532–1543, 2014.

[10] M. Peters, M. Neumann, M. Iyyer and M. Gardner, "Deep contextualized word representations," *arXiv preprint*, no. 1802, 2018.
[11] J. Schmidhuber and S. Hochreiter, "Long short-term memory," *Neural computation*, no. 9, pp. 1735–1780, 1997.
[12] M. Schuster and K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE Transactions on Signal Processing*, no. 11, pp. 2673–2681, 1997.
[13] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv*, no. 1810, 2018.
[14] L. Zhenzhong, C. Mingda, G. Sebastian, G. Kevin, S. Piyush and S. Radu, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv*, pp. 1909–11942, 2019.
[15] L. Yinhan, O. Myle, G. Naman and D. Jingfei, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
[16] S. Victor, D. Lysandre and C. Julien, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.
[17] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan and P. Dhariwal, "Language models are few-shot learners," *Advances in neural information processing systems*, no. 33, pp. 1877–1901, 2020.
[18] B. Peng, C. Li, P. He, M. Galley and J. Gao, "Instruction tuning with GPT-4," *arXiv preprint arXiv:2304.03277*, 2023.
[19] R. Al-Rfou, B. Perozzi and S. Skiena, "Polyglot: Distributed word representations for multilingual nlp," *CoNLL*, pp. 183–192, 2013.
[20] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, no. 5, pp. 135–146, 2017.
[21] A. Hadifar and S. Momtazi, "The impact of corpus domain on word representation: a study on Persian word embeddings," *Language Resources and Evaluation*, no. 52, pp. 997–1019, 2018.
[22] M. Zahedi, M. Yadollahi, H. Bokaei and E. DoostMohammadi, "Persian word embedding evaluation benchmarks," *IEEE*, pp. 1583–1588, 2018.
[23] M. Farahani, M. Farahani, M. Gharachorloo and M. Manthouri, "Parsbert: Transformer-based model for persian language understanding," *Neural Processing Letters*, pp. 3831–3847, 2021.
[24] S. Bostan, A. ZareBidoki and M. Pajoohan, "Semantic word embedding using BERT on the Persian web," *Iranian Journal of Electrical and Computer Engineering*, 2023.

[25] L. Shaohua, C. Tat-Seng, Z. Jun and C. Miao, "Generative Topic Embedding: a Continuous Representation of Documents (Extended Version with Proofs)," *arXiv preprint arXiv:1606.02979*, 2016.
[26] B. Mitra, E. Nalisnick, N. Craswell and R. Caruana, "A Dual Embedding Space Model for Document Ranking," in *25th International Conference Companion on World Wide Web*, 2016.
[27] M. Dehghani, H. Zamani, A. Severyn and J. Kamps, "Neural ranking models with weak supervision," in *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.
[28] C. Xiong, Z. Dai and J. Callan, "End-to-end neural ad-hoc ranking with kernel pooling," in *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.
[29] R. Brochier, A. Guille and J. Velcin, "Global vectors for node representations," in *In The World Wide Web Conference*, 2019.
[30] A. Gourru and J. Velcin, "Gaussian embedding of linked documents from a pretrained semantic space," in *In Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 2021.
[31] R. Menon, J. Kaartik and K. Nambiar, "Improving ranking in document based search systems," in *In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2020.
[32] J. Li, C. Guo and Z. Wei, "Improving Document Ranking with Relevance-based Entity Embeddings," in *In 2022 8th International Conference on Big Data and Information Analytics (BigDIA)*, 2022.
[33] S. Han, X. Wang, M. Bendersky and M. Najork, "Learning-to-Rank with BERT in TF-Ranking," Google Research Tech Report, 2020.
[34] V. Gupta, A. Dixit and S. Sethi, "A Comparative Analysis of Sentence Embedding Techniques for Document Ranking," *Journal of Web Engineering*, pp. 2149–2186, 2022.
[35] R. Mihalcea, C. Corley and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *in the 21st National Conference on Artificial Intelligence*, 2006.
[36] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, no. 20, pp. 422–446, 2002.
[37] S. Qu, Y. Zhang, Y. Ji and Z. Wang, "Online-Review-Driven Products Ranking: A Hybrid Approach," *Systems*, p. 11(3), 2023.

[38] Q. Lyu, K. Chakrabarti and S. Hathi, "Hybrid Ranking Network for Text-to-SQL," *arXiv preprint arXiv*, 2020.

[39] I. Alghanmi, L. Espinosa-Anke and S. Schockaert, "Combining BERT with static word embeddings for categorizing social media," in *In Proceedings of the sixth workshop on noisy user-generated text (w-nut 2020)*, 2020.

Biographies

Shekoofe Bostan is a Ph.D. candidate in computer engineering at Yazd University in Iran. She is now a lecturer in the university's computer engineering department and a software developer at a leading cloud search firm. Her research focuses on deep learning, semantic information retrieval, and semantic analysis of social networks.

Ali Mohammad Zareh Bidoki obtained his Ph.D. in software engineering at the Electrical & Computer Engineering department of University of Tehran. He also obtained a master's degree from the ECE department of University

of Tehran and Bachelor's degree from the Computer Engineering department of Isfahan University of Technology. He is currently an associate professor in the CE department of Yazd University and CEO of Parsijoo (a search engine company). His expertise is information retrieval and search engines.

Mohammad-Reza Pajoohan is currently Assistant Professor in the department of Computer Engineering of Yazd University, Yazd, Iran. He got his Ph.D. from the department of Computer Science of Universiti Sains Malaysia (USM) and National University of Singapore (NUS). He graduated with M.Sc. and B.Sc. degrees in computer engineering from Sharif University, Tehran, Iran. His research interests including database, data mining, data science and privacy preserving data publication.