Dean&Francis
ISSN 2959-6149

Exploring Vector-Based Methods for Effective Document Retrieval

Xian Zhang

Abstract:
With the development of online storage services and the internet, people, especially students, often collect numerous documents from the servers. Searching from abundant documents can become a difficult task. With the rapid development of the LLM (Large Language Model) and word embedding technique, people should have the chance to find the information in a new way, which is much more effective. In this paper, with the vectorization of documents and users' questions with different models, they can somehow understand the words of users and find the correspond documents which the users want. And LLM can directly extract the important information in the documents. Therefore, users can easily find the information they want from a large number of documents.. This paper provides four methods with three main models to convert documents to vectors. The best one can retrieve more than 90% documents in the testing data set with the given keywords.

Keywords: Vector Searching; Document Indexing; Semantic Search; LLM; Word Embedding; Information Retrieval

1. Introduction

The history of search engines dates to 1990, when students at the University of McGill invented Archie, a tool designed to index FTP documents. However, users had to type the exact file name into Archie to retrieve the desired file. Although the search engines are very advanced nowadays. The main searching principle remains the same: they are mostly based on keywords.

A significant limitation of keyword-based search engines is that users often have difficulty finding the documents they need because of the need for precise keywords. Users may have difficulty accessing specific information in a document that they are actually looking for, and users may not know the keywords required to find the information they need. Or they use words with similar meanings to express the same idea. For example, if someone wants to search for the word "queen" but cannot remember the word, he may not find relevant results.

Vector search offers a solution to this problem by calculating the meaning of the searching request and can do the calculations of words by the calculation of word vectors. For instance, if someone forgets the word "queen" but remembers related words like

ISSN 2959-6149

"male," "king," and "female," they can still find "queen" through two calculations: "king" + "female" = "queen" and "king" - "male" = "queen." Also, this is a kind of method to search the information without using the exact keyword. For example, if you want to search "stomach", you can type in "the digestion system". Therefore, vector search provides a new way to search based on word meanings.

This paper proposes a method to build indexes for network file servers by vectorizing document features partially. This reduces the size of the indexes and retains essential information needed from the documents. By learning keyword relationships, vector search can enhance search results for similar meanings and reduces search time, resulting in a more efficient and effective document retrieval method.

In vector search, search queries and documents are both converted into vectors, enabling the computer to understand the intent behind the queries and locate documents within vast datasets by calculating the similarity between the queries' vectors and the documents' vectors. Models like Doc2vec (Document to Vector), Word2vec (Word to Vector), and BERT (Bidirectional Encoder Representations from Transformers) are used to vectorizing documents.[12] And the algorithm of TF-IDF (Term Frequency-Inverse Document Frequency) is utilized to identify the important words in documents. [8]By changing the combinations of models and methods, we determine the most accurate one with a recall rate exceeding 90%.

We assess the performance of the models using the dataset containing academic papers and keywords employing IR metrics such as the recall rate. Our research endeavors if vector-based search techniques can be used in document searching and make a comparation between different methods.

What's more, by integrating an LLM with our method, users can input nonstandard queries, like keywords or non-straightforward questions. As long as the query isn't so complex, our method can be utilized with LLM and output a specific answer to users' queries.[11] With LLMs, you can even find exact sentences and have it provided a summary without encountering the 'LLM hallucination'.

2. Literature Review

2.1 Distributed representation based neural network (word vectors)

Due to the 'Semantic Gap', computers have difficulties to distinguish the similarity between the words such as 'mic' and 'microphone'. Therefore, there is a method to convert the words or concepts to high-dimensional vectors to solve this problem. [13]Through this method, similar words or concept will be close to each other in the vector space. [9]And computers will be able to understand the words' relationship. This is an important work in NLP. And there are two main methods to convert words into vectors. One is the one-hot representation. This method is the most common. It will give every word a single 'ID' vector to represent the words. For example: 'apple'[1,0,0,0,0,0,0], 'fruit'[0,1,0,0,0,0,0]. This is enough to represent a word, but lack of the meaning of the word. And the vector depth will also be very large if the vocabulary is large.

Therefore, a second method of vectorizing words called the distributed representation appears.[10] Unlike the former method. This method can contain the meaning of the words. This method is based on the distributional hypothesis, which was given by Harris in 1954[1], says that words with similar contexts also have similar semantics. Based on this hypothesis, researchers have proposed a variety of word representation models, such as the matrix-based LSA model [2], the clustering-based Brown clustering model [3], and the neural network word representation model which this paper focuses on. And the vectors generated by the neural networks are often called the word embedding(vector). For documents, the word vectors are not enough, so we need to get the vector of documents with different methods.

Neural network word vector representation technology uses neural network technology to model context and the relationship between context and target word. Since neural networks are more flexible, the biggest advantage of this type of method is that it can represent complex contexts. In the previous matrix-based distribution representation method, the most commonly used context is words. If n-grams containing word order information are used as contexts, when n increases, the total number of n-grams will increase exponentially, and the dimensionality curse problem will be encountered. When neural networks represent n-grams, they can combine n words in some combinations, and the number of parameters only grows at a linear rate. With this advantage, neural network models can model more complex contexts and include richer semantic information in word vectors.[4]

2.2 Word2vec model

Word2vec is a model in natural language processing (NLP) for getting the vector of the words in a group of documents. [5] These vectors capture information about the meaning of the word based on the surrounding words.

Similar words tend to have the same vector values and are grouped in the same block, which can be seen in Figure1. And Word2Vec can get the word vectors after training of a large corpus. The resulting similarity value is obtained from the word vector value than calculated using the Cosine Similarity equation. The similarity value produced by Word2Vec ranges from -1 to 1 as the highest similarity value.[14]

Word2vec is based on Continuous Bag of Words Model(CBOW) and Skip-Gram Model.

CBOW represents a statistical language model that predicts the likelihood of a words occurrence by considering the preceding C words or the C surrounding words.

The Skip-Gram Model, on the other hand, calculates the probability of a word appearing before and after it based on a certain word.[19]

Word2vec is an efficient framework of architectural CBOW and Skip-Gram to calculate vector representations of words. The CBOW and Skip-Gram architectures can be seen in figure2.[14]

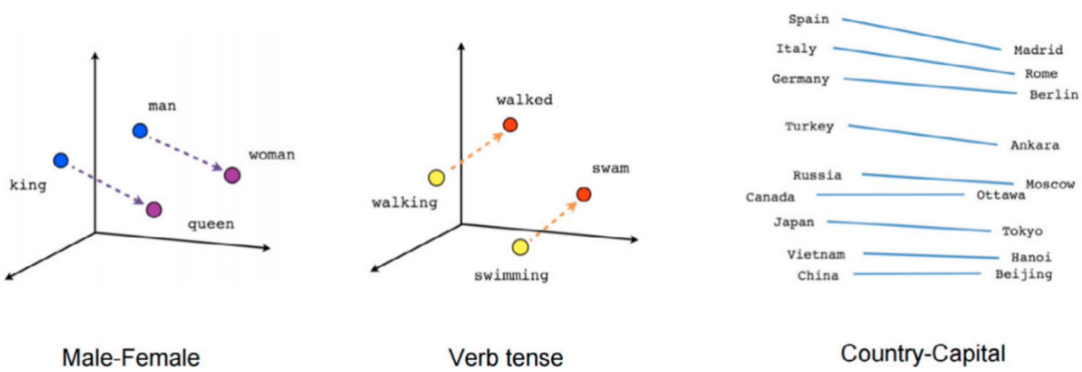


Figure 1. Word2Vec representation

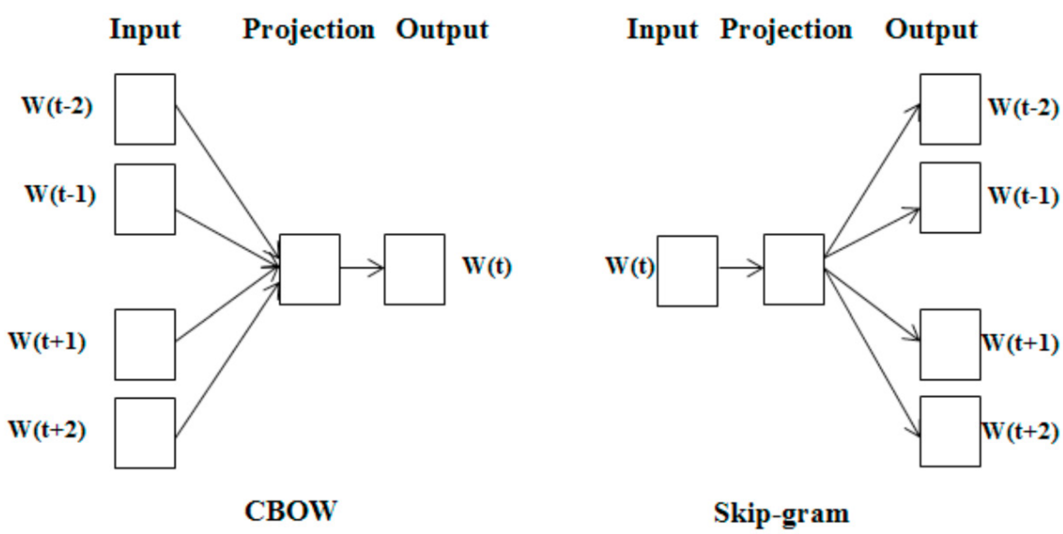


Figure2. Word2Vec CBOW and Skip-Gram models architecture

2.3 Doc2vec

Doc2Vec is a neural network which learns the distributed representation of documents. Le and Mikolov in 2014 introduced the Doc2Vec algorithm, which usually outperforms such simple averaging of Word2Vec vectors.[6]

The basic idea is act as if a document has another floating word-like vector, which contributes to all training predictions, and is updated like other word-vectors, but we will call it a doc-vector. Gensim's Doc2Vec class implements this algorithm.

There are two implementations.

2.3.1 Paragraph Vector - Distributed Memory (PV-DM)

PV-DM is analogous to Word2Vec CBOW. The doc-vectors are obtained by training a neural network on the synthetic task of predicting a center word based an average of both context word-vectors and the full document's doc-vector.

2.3.2 Paragraph Vector - Distributed Bag of Words (PV-DBOW)

PV-DBOW is analogous to Word2Vec SG. The doc-vec-

ISSN 2959-6149

tors are obtained by training a neural network on the synthetic task of predicting a target word just from the full document's doc-vector.

2.4 Bert

BERT is a pre-trained language model based on Transformer, aimed at generating bidirectional representations of text through deep learning techniques. During the pre-trained period, BERT learns language representations through two main tasks, namely Masked Language Model (MLM) and Next Sentence Prediction (NSP). The MLM task involves randomly masking some words in the input sentence and having the model predict these masked words. The NSP task is to predict whether two sentences are continuous.[15] After pre training is completed, the BERT model can be fine-tuned for specific tasks such as text classification, QA systems, named entity recognition, etc.

2.5 TF-IDF

In information retrieval, term frequency–inverse document frequency(TF-IDF) is a measurement to show how important a word is in a group of documents. It is the product of term frequency(TF) and inverse document frequency(IDF). [20]

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

The Inverse Document Frequency (IDF) is designed to measure the term-specificity.[16] Document Frequency (DF) is how many times a word appears in a group of documents. And IDF can be calculate as

$$IDF(t) = log\left(\frac{N}{1+df(t)}\right)$$

Therefore, IDF can be used to measure how much the information a word can represent in a document. TF is term frequency which equation is

$$TF(t,d) = \frac{f_{t,d}}{\Sigma_{t'} \in d^{f} t', d}.$$

Where, $f_{t,d}$ is the frequency of word *t* in the document *d,* and $\Sigma_{t'} \in d^{f} t', d$ is the number of words in document *d.*

2.6 MS MARCO models

MS MARCO (Microsoft Machine Reading Comprehension) is a data set provided by Microsoft to promote the research and development of machine reading comprehension and question-answering systems. This data set focuses on real-world information retrieval scenarios, including document retrieval and retrieval-style dialogue. The data set consists of 1,010,916 anonymous questions - sampled from Bing's search query logs - each with a human-generated answer and 182,669 fully human-generated answers.
The msmarco-distilrobert-base-v4, msmarco-distilrbert-cos-v5 and msmarco- distilroberta-base-v2 models are versions of the DistilBERT model optimized for this data set. These models are fine-tuned on the MS MARCO data set to improve performance in question answering tasks.

2.7 Cosine Similarity

Cosine Similarity is a metric used to measure the similarities between two vectors. In this paper, this method is used to calculate the similarities between documents and keywords, to show the similarities between the keywords and documents. Unlike Euclidean distance, cosine Similarity focuses on the direction of the vector rather than its length. It is calculated as follows:

$$CosineSimilarity = \frac{A \cdot B}{|A||B|}$$

Where, $A \cdot B$ is the dot product of vectors, and $|A||B|$ is the norms (usually Euclidean norms) of vectors.
The value of cosine similarity is between -1 and 1. The closer the value is to 1, the more similar the two vectors are; the closer the value is to -1, the less similar the two vectors are.

3. Methodology

3.1 The preprocess of documents

3.1.1 Documents preprocessing

In this paper, all documents are processed using the following methods.

Step1. Traverse all PDFs under the folder.
Step2. Do Optical Character Recognition (OCR) for the PDFs
Step3. Split some texts which are too long to several pieces.
Step4. Store them in a csv file with format: [File Path], [Text]
Step5. Delete some unnecessary words. Such as stopwords, and store the processed text in another column with format: [File Path], [Text],[Original Text], [Processed Text]

3.1.2 Testing data set and recall rate calculating

3.1.2 .1 Data set preprocessing

In this paper, a method is needed to evaluate the performance of different methods. And the data set 'Nguyen2007 dataset' is used as the testing dataset. In this dataset, there are 211 documents, and the mean length of documents is 7032.616 words. There are TXT files and KWD files in it. The TXT files are the contents of some papers, and the KWD files contain the keywords for these documents, which are concluded by human. Each KWD file has the same filename as its corresponding TXT file. The process method of the testing dataset is almost the same as the previous method of processing PDF files. But TXT files can be directly stored, therefore, the detailed method of preparing the dataset is:

Step1. Split some texts which are too long to several pieces.
Step2. Store them in a csv file with format: [File Path], [Text]
Step3. Delete some unnecessary words. Such as stopwords, (resources from www.nltk.org) and store the processed text in another column with format: [File Path], [Text], [Original Text], [Processed Text]
Step4. Store the keywords from the KWD files and the file path of corresponding TXT files in another csv file with format: [File Path], [Keywords]

Then, methods will be tested by finding the txt files with the keywords.

3.1.2.2 Recall rate calculating

There's a program to convert the searching requests to vectors, enable the model to calculate the documents most similar to the keywords with cosine similarity. I calculate the recall rates with this program with different Top-N(The top N documents most likely to be the result)

$$R(recallrate) = \frac{The\ number\ of\ the\ finded\ documents}{The\ number\ of\ the\ total\ documents}$$

And this parameter can show the possibility of the model to find the correct document. For example, the recall rate is 0.7, then the model will have the possibility of 0.7 to find the correct document.

3.2 Vectorizing Methods

3.2.1 Doc2vec

In this method, doc2vec is the only model. The model is trained with the processed text in the csv file and the file paths performs as the tags. Different combinations of parameters are used, and the models are visualized to some 2-dimentional graphs with T-SNE(Figure3) and UMAP(-Figure4)

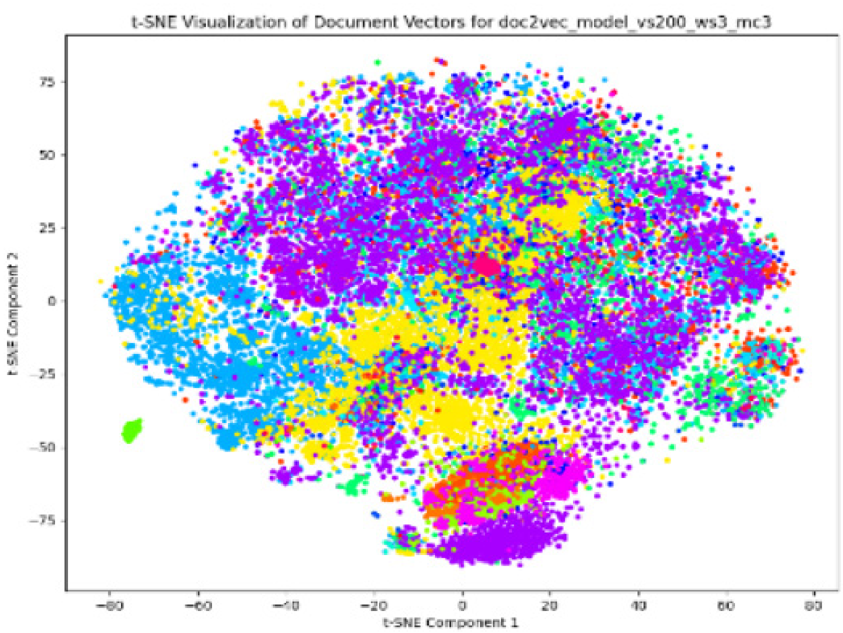


Figure3. t-SNE doc2vec_vs200_ws3_ms3

ISSN 2959-6149

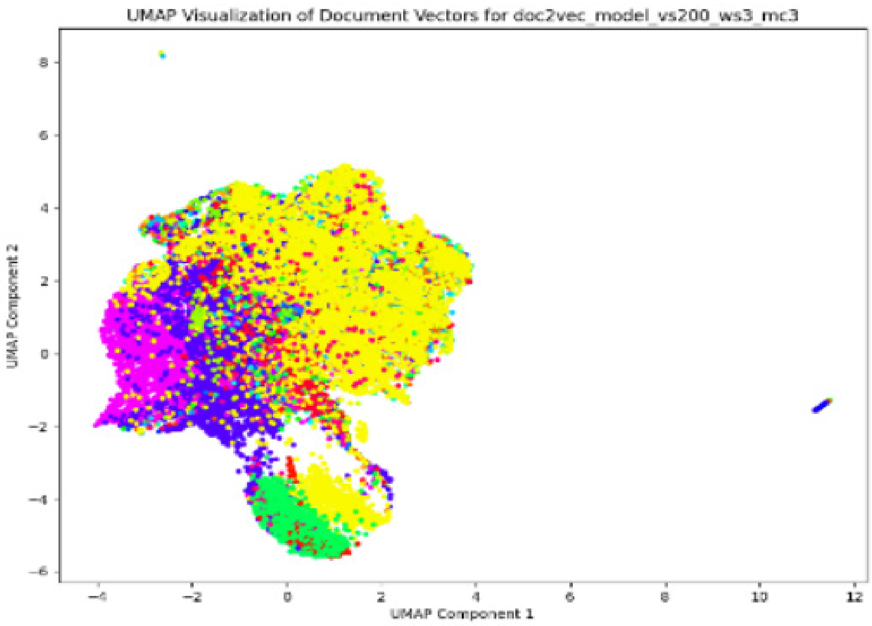


Figure4. UMAPdoc2vec_vs200_ws3_ms3

Each color represents a different document, and the dots are the text slices. The graphs shows that although the dots seem to have a tendency to aggregate, but still mixed together, the situations are almost the same in other combinations of parameters. And this means that the vectors of different documents are mixed, which implies that it may not result in an effective search method.

3.2.2 Word2vec

3.2.2 .1 No pre-trained model

This method doesn't use the pre-trained models. It uses the processed text and tags in the csv file. And the document vector is calculated by the following method:

Step1. Use the word2vec CBOW method to calculate the word vectors in the documents.
Step2. Collect all the word vectors in each document and calculate the mean vector of them to represent the document vectors.
Step3. While querying the models, the questions are processed the same way as 2 to get the question vector.
Step4. The similarity of the questions and documents are calculated to find the most relative documents with cosine similarity.

3.2.2 .2 Pre-trained model

This method uses several pre-trained models of word2vec. And use the processed text and tags in the csv file. And the document vector is calculated by the following method:

Step1. Use the pre-trained models to calculate the word vectors in the document when a word exists in the vocabulary of the models.
Step2. Collect all the word vectors in each document and calculate the mean vector of them to represent the document vectors.
Step3. While querying the models, the questions are processed the same way as Step2 to get the question vector.
Step4. The similarity of the questions and documents are calculated to find the most relative documents with cosine similarity.

3.2.3 Bert+TF-IDF

This method utilizes the model 'bert-base-nli-mean-tokens'. This is a sentence-transformers model that maps words to a 768-dimensional dense vector space, making it suitable for tasks such as clustering or semantic search. This paper employs TF-IDF to identify the important words within a document and stores their vectors along with the file path. During the search process, BERT can determine the most similar words to the user's searching request from the saved word map and locate the related documents containing words that are close in vector space. The specific steps are as follows:

1. Calculating the Word Vectors

Step1. Remove the stopwords and use TF-IDF to identify the most important words (10 words in this experiment) in the documents. Step2. Calculate the vectors for these words using the model 'bert-base-nli-mean-tokens'. Step3. Store the word vectors in a CSV file, along with the file path, words, and their TF-IDF values.

2. Searching the Documents

Step1. Remove the stopwords and calculate the vectors for the input questions. Step2. Find similar words in the previously stored CSV file by calculating the vector similarities. Step3. Output the files related to the similar words.

Results are also tested with the testing dataset.

3.2.4 MS MARCO Models

This method uses 3 pretrained model which are trained based on MS MARCO dataset. They are msmarco-distilrobert-base-v4, msmarco-distilrbert-cos-v5 and msmarco-distilroberta-base-v2.

The models are trained with the processed text in the csv file and the file paths performs as the tags. And cosine similarities are used to query the models as the doc2vec model.

4. Results

4.1 Doc2vec

The performance of doc2vec is shown below in Figure5.

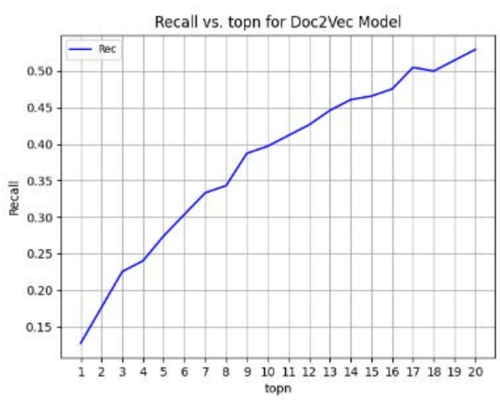


Figure5. Recall vs. topn for Doc2vec Model

The recall rate is about 0.5 when Top-N is 17(Figure5), which is really a bad score. I think this is because the lack of training data. For doc2vec has difficulty when the training data is very little. A pre-trained model may solve this problem. But there are few pre-trained doc2vec model. Therefore, this method is abandoned.

4.2 Word2vec

The performance of word2vec CBOW without pre-trained model is shown below.

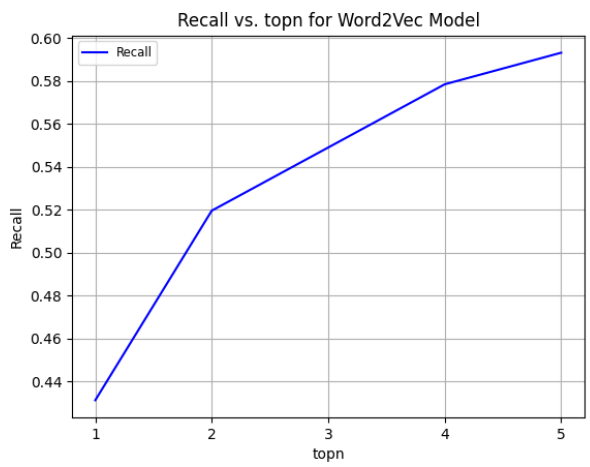


Figure6. Recall vs. topn for Word2vec Model without pre-trained model.

The recall rate for Word2vec Model without pre-trained model is about 0.593 when Top-N is 5(Figure6), which is better than Doc2vec.

However, the recall rate with pre-trained model(word2vec-google-news-300) is about 0.8 when Top-N is 5(Figure7), which is much more better than other models.

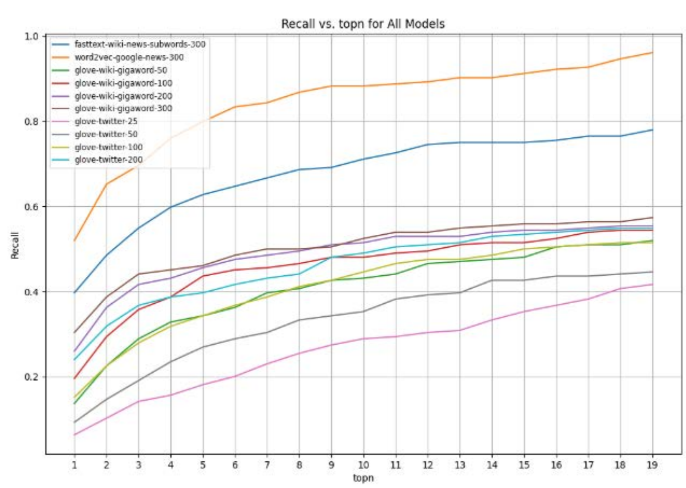


Figure7. Recall vs. topn for Word2vec Model with pre-trained model.

ISSN 2959-6149

4.3 TF-IDF+BERT

The performance of TF-IDF with BERT is shown below.

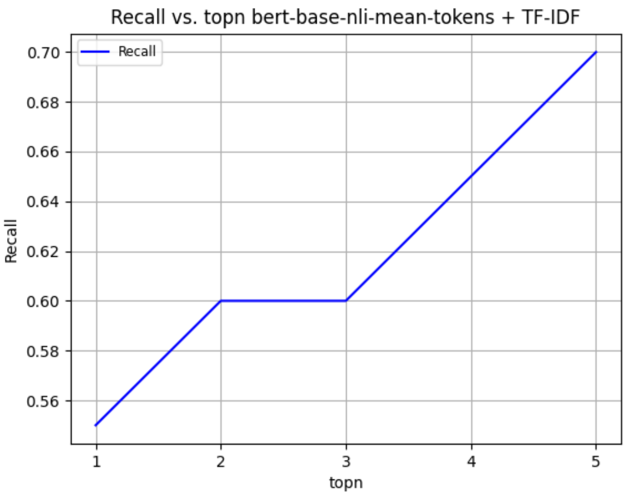


Figure8. Recall vs. topn for bert-base-nli-mean-tokens + TF-IDF

The recall rate is about 0.7 when Top-N is 5(Figure8), which is lower than pre-trained word2vec.

4.4 MS MARCO

Below is the result of this method.

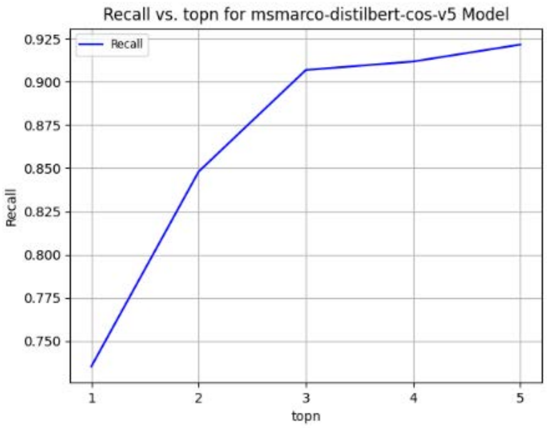


Figure9. Recall vs. topn for msmarco-distilroberta-base-v2 model

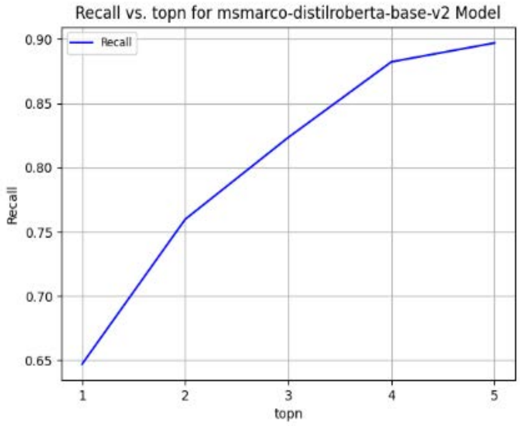


Figure10. Recall vs. topn for msmarco-distilrbert-cos-v5 model

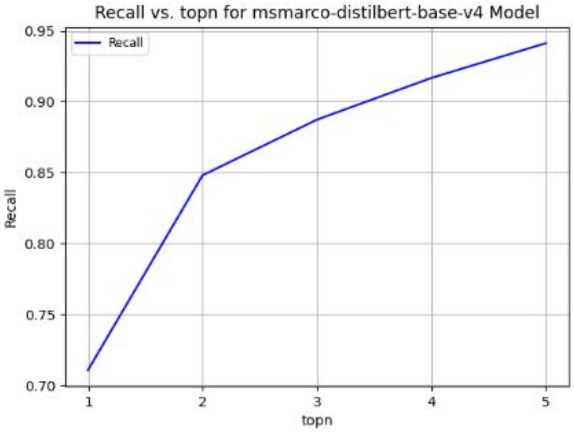


Figure11. Recall vs. topn for msmarco-distilrobert-base-v4 model

The recall rate is the highest in msmarco-distilrobert-base-v4 model, it's about 0.94 when Top-N is 5(Figure11). And a little lower in msmarco-distilroberta-base-v2 model and msmarco-distilroberta-cos-v5 model, with recall rate at 0.90 and 0.925 when Top-N is 5(Figure9, Figure10).

5. Discussion

This study finds that by vectorizing the documents, computers can understand the content of the document effectively. Being the same as the findings of Smith et al. (2020), our study also shows that some deep learning models performs well in word embeddings in handling complex queries. And the msmarco-distilbert-base-v4 model performs the best, with a recall rate increasing by 30% compared to the traditional word2vec model. With topn=5, the recall rate can be up to 92.5%. Which means that this method can be a very effective way to recall the documents from abundant of them. However, a major limitation of this study is the relatively small sample size. And methods are only tested on one dataset, which may not show accurate differences between the methods. Additionally, our research focused only on English documents, therefore, other languages still need to be confirmed. In the companies' Q&A system, or some law documents retrieval, our method of vectorizing the documents can greatly enhance retrieval efficiency. With the advancement of large language models (LLMs), future research could explore integrating these models with our vectorization techniques to further improve performance and handle a broader range of queries.

6. Conclusion

The msmarco-distilbert-base-v4 model demonstrated the highest performance, achieving a recall rate of up to 92.5% with topn=5, representing a 30% and more improvement over traditional word2vec models. Therefore, it's very effective in retrieving information from abundant documents. With the combination of LLM, [Appendix III] users can easily retrieve the exact document by asking some verbal-like questions. And this combination brings the information retrieval field to a vector&LLM era.

References

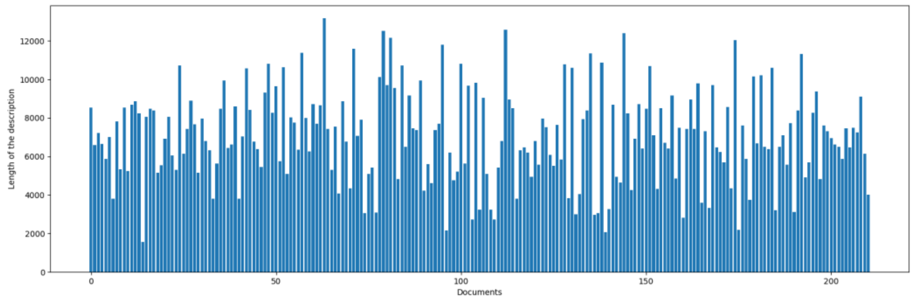
[1] Harris, Z. S. (1954). Distributional structure. Word, 10(23), 146-162.
[2] Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. Discourse Processes, 25(2-3), 259–284.
[3] Brown, P. F., Desouza, P. V., Mercer, R. L., Della Pietra, V. J., & Lai, J. C. (1992). Class-based n-gram models of natural language. Computational Linguistics, 18(4), 467–479.
[4] 来斯惟 . (2016). 基于神经网络的词和文档语义向量表示方法研究 (Doctoral dissertation, 北京 : 中国科学院大学).
[5] Church, K. W. (2017). Word2Vec. Natural Language Engineering, 23(1), 155- 162.
[6] Le, Q., & Mikolov, T. (2014,). Distributed representations of sentences and documents. In Proceedings of the International Conference on Machine Learning (pp. 1188-1196). PMLR.
[7] Zhu, W., Wu, X., & Chen, J. (2019). Spatial keyword query of region-of-interest based on the distributed representation of point-of-interest. ISPRS International Journal of Geo-Information, 8(6), 287.
[8] Yucel, O., & Sen, S. (2018). Language independent recommender agent. The Knowledge Engineering Review, 33, e15.
[9] de Paula, L. B., Villaça, R. D. S., & Magalhães, M. F. (2011). Organization of multimedia data for conceptual search based on ontologies. Journal of the Brazilian Computer Society, 17, 241-254.
[10] Jiang, F., Liu, Y. Q., Luan, H. B., Sun, J. S., Zhu, X., Zhang, M., & Ma, S. P. (2015). Microblog sentiment analysis with emoticon space model. Journal of Computer Science and Technology, 30(5), 1120-1129.
[11]Moallim, A. A., & Yun, L. J. (2021). Structural Application of Medical Image Report Based on Bi-CNNs-LSTM-CRF. In Advances in Computer, Communication and Computational Sciences: Proceedings of IC4S 2019 (pp. 365-377). Springer Singapore.
[12] Xiong, Y., Chen, S., Qin, H., Cao, H., Shen, Y., Wang, X., ... & Tang, B. (2020). Distributed representation and one-hot representation fusion with gated network for clinical semantic textual similarity. BMC medical informatics and decision making, 20, 1-7.
[13] de Paula, L. B., Villaça, R. D. S., & Magalhães, M. F. (2011). Organization of multimedia data for conceptual search based on ontologies. Journal of the Brazilian Computer Society, 17, 241-254.
[14] Jatnika, D., Bijaksana, M. A., & Suryani, A. A. (2019). Word2vec model analysis for semantic similarities in english words. Procedia Computer Science, 157, 160-167.
[15] Alzubi, J. A., Jain, R., Singh, A., Parwekar, P., & Gupta, M. (2023). COBERT: COVID-19 question answering system using BERT. Arabian journal for science and engineering, 48(8), 11003-11013.
[16] Pant, G., & Srinivasan, P. (2010). Predicting web page status. Information Systems Research, 21(2), 345-364.
[17] OpenAI. (2024). ChatGPT (Version GPT-4) [Large language model].
[18] Kimi. (n.d.). Kimi intelligent assistant [Online].
[19] Ma, L., & Zhang, Y. (2015, October). Using Word2Vec to process big text data. In 2015 IEEE International Conference on Big Data (Big Data) (pp. 2895-2897). IEEE.
[20] Aizawa, A. (2003). An information-theoretic perspective of tf–idf measures. Information Processing & Management, 39(1), 45-65.

Appendixces I

Nguyen:2007
@InProceedings{Nguyen:2007,
author = {Thuy Dung Nguyen and Min-Yen Kan},
title = {Key phrase Extraction in Scientific Publications},
booktitle = {Proceeding of International Conference on Asian Digital Libraries},
year = {2007},
pages = {317--326}

ISSN 2959-6149

}



Appendix II

Models of MS MARCO

Approach	NDCG@10 (TREC DL 19 Re-ranking)	MRR@10 (MS Marco Dev)	Queries (GPU / CPU)	Docs (GPU / CPU)
Models tuned for cosine-similarity				
msmarco-MiniLM-L-6-v3	67.46	32.27	18,000 / 750	2,800 / 180
msmarco-MiniLM-L-12-v3	65.14	32.75	11,000 / 400	1,500 / 90
msmarco-distilbert-base-v3	69.02	33.13	7,000 / 350	1,100 / 70
msmarco-distilbert-base-v4	70.24	33.79	7,000 / 350	1,100 / 70
msmarco-roberta-base-v3	69.08	33.01	4,000 / 170	540 / 30
Models tuned for dot-product				
msmarco-distilbert-base-dot-prod-v3	68.42	33.04	7,000 / 350	1100 / 70
msmarco-roberta-base-ance-firstp	67.84	33.01	4,000 / 170	540 / 30
msmarco-distilbert-base-tas-b	71.04	34.43	7,000 / 350	1100 / 70

Appendix III

Combination with LLM
GPT-4o:[17]
User:
What is IEEE 802.11 in Bluetooth?
GPT-4o:
File: ../data/all\45.txt
Sentence: “This same frequency band is already in use by microwave ovens and the popular Wireless Local Area Network (WLAN) devices implementing the IEEE 802.11 standard specifications [8].”
Answer: IEEE 802.11 refers to the set of standards for Wireless Local Area Network (WLAN) devices, which share the same 2.4 GHz ISM band as Bluetooth devices.
moonshot-v1-8k: [18]
User:
Wireless technology
Kimi:
File: ../data/all\176.txt
Sentence: “Recent advances in wireless networking technology, low-power digital circuits, sensing materials and Micro Electro-Mechanical Systems (MEMS) opened up the possibility of building small sensor devices capable of data processing, remote sensing and wireless communication.”
Answer: The development in wireless technology, along with advancements in low-power digital circuits, sensing materials, and MEMS, has enabled the creation of compact sensor devices. These devices are capable of performing data processing, remote sensing, and wireless communication, which are crucial for the operation of wireless sensor networks.