ICIC Express Letters
Part B: Applications
Volume **15**, Number **4**, April **2024**

ICIC International ©2024 ISSN 2185-2766
pp. **373–380**

FAST PASSAGE RETRIEVAL IN WEIGHTED HAMMING SPACE FOR OPEN-DOMAIN QUESTION ANSWERING

Richeng Xuan[1,3], Junho Shim[2,*] and Sang-goo Lee[3]

[1]Beijing Academy of Artificial Intelligence (BAAI)
Beijing 100080, P. R. China
rcxuan@baai.ac.cn

[2]Department of Computer Science
Sookmyung Women's University
Cheongpa-ro 47-gil 100 (Cheongpa-dong 2ga), Yongsan-gu, Seoul 04310, Korea
*Corresponding author: jshim@sookmyung.ac.kr

[3]Department of Computer Science and Engineering
Seoul National University
1 Gwanak-ro, Gwanak-gu, Seoul 08826, Korea
sglee@europa.snu.ac.kr

Received July 2023; accepted September 2023

Abstract. *The latest research has shown that hashing can decrease memory usage and computing time in many applications without significantly reducing efficiency. Recently, hashing approaches have been used in open-domain question answering tasks, where only the most basic hashing layer is employed. We propose a weighted Hamming distance-based semantic hashing method that learns optimal dimension weights to obtain effective passage retrieval results. We compared the performance of our method to those of state-of-the-art semantic hashing baselines in a passage retrieval task on open-domain question answering. The results show that our method can outperform the baselines and achieve accuracy similar to that attainable with dense vector-based retrieval.*
Keywords: Natural language processing, Passage retrieval, Semantic hashing, Hamming distance, Weighted Hamming, Open-domain question answering

1. **Introduction.** Open-domain question answering (QA) [1] is a task in which factual questions are answered using a large collection of documents (e.g., Wikipedia). Recent QA systems often use a simplified two-stage approach [2]. Firstly, a *retriever* is utilized to retrieve a small number of documents from a large collection and a *reader* is used to select the correct answer from a small number of retrieved documents. We also refer to this method as the *retriever-reader* approach. Although the accuracy of the reader obtaining the answers from the candidate is sufficiently high, it is impossible to use the reader to calculate all the data in a very large collection. Therefore, using a less computationally complex retrieval to generate, small candidates for readers become realistic and feasible.

Owing to the breakthrough regarding the pre-trained language model [3], most natural language processing (NLP) tasks represent documents as continuous values. Recent retriever approaches [4,5] also encode questions and passages using pre-trained encoders to obtain continuous vectors. In these methods, the relevant passages are retrieved by performing a similarity search on the index containing all of the large collection passage embeddings with a question embedding as a query. Retrieval using pre-trained embedding often outperforms classical methods, but as the pre-trained language model is developed further, the document structure embedding becomes more complex. This situation directly causes the embedding of a large collection to be very large. For example, the continuous

DOI: 10.24507/icicelb.15.04.373

embedding of common knowledge sources (e.g., Wikipedia) requires 65 GB of memory to store [6].

Semantic hashing [7,8] methods enable very efficient searching by learning to represent documents as compact binary vectors called hash codes. By using a compact binary hash code, memory usage and computational cost can be reduced. The binary passage retriever (BPR) [6] integrates hashing into a state-of-the-art dense passage retriever (DPR) [4] to learn binary hash code from original continuous vectors, which significantly reduces the size of the passage index of a large collection. However, the BPR performs worse than DPR in some cases. BPR reduces the memory cost with loss of accuracy on $k < 20$.

The previous hashing approaches have often used the Hamming distance of the binary code as the similarity index, but the similarity that can be obtained by simply counting the Hamming distance of different dimensions is at most equal to the number of dimensions. This situation also leads to a large amount of similarity in large-scale texts. Usually, the number of passages is more than several million. If only a few hundred bits of hash code are used, the similarity of the query is at least the same as that of thousands of passages. Weighted hashing is one solution to this problem [9], and the weight of each dimension will make the similarity less discrete. In the latest hashing research [10,11], scholars began to use the weight of each dimension to improve the accuracy of hash code extraction and accelerate partial retrieval.

This paper proposes a weighted semantic hashing method for passage retrieval by using the dimension weight to achieve accuracy similar to that attainable by dense vector-based retrieval for the first time. Further, it presents experiments conducted on the benchmark dataset from a previous study and a demonstration that the proposed weighted semantic hashing outperforms the existing hashing approaches. In addition, it describes the optimization of the learning method based on the dimension weight and presents a learning technique to improve the training effect.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the proposed model. Section 4 presents the experimental results and corresponding discussions. Finally, Section 5 concludes the paper.

2. **Related Work.**

Related passage retrieval is an important component of open-domain QA [1,12]. It is difficult to obtain answers efficiently using the answer extraction component, and passage retrieval significantly reduces the search space for answer extraction. From the beginning, the sparse vector model using bag-of-words (BOW) has been widely employed as a passage retriever. TF-IDF [13] and BM25 [14] improve the retrieval accuracy and make the sparse vector model a standard method applied to many QA tasks [2,15]. The limitation of BOW is that this method can only be based on keyword matching and lacks a search based on semantics.

To reduce this limitation, recent researchers [16] have begun to use external information to reduce this limitation further. Compared with research on the sparse model, studies using dense vectors started very early, and latent semantic analysis [17,18] was the first dense vector model to be applied. Owing to the development of neural networks, the DPR [4] using the pre-trained model [3] has gradually emerged, relying on the innate advantages of the pre-training model in terms of general language semantics, which is also used in the state-of-the-art QA method [19]. Because the DPR model uses a high-dimensional dense vector, when dual with more than 10 million passage vectors, it requires a considerable amount of memory, which is not affordable for ordinary computers. To reduce the memory cost, BPR [6] employs the learning-to-hash method to use binary code to store the passage vector.

The original text data can be represented using compact binary codes through hashing. The objective of hashing is to reduce the memory and search time cost of the nearest neighbor search by representing data points using compact binary codes. In recent years,

with the development of semantic hashing, the solution of binary code is no longer a last resort, entailing considerable loss of precision to achieve efficiency. Variational deep semantic hashing [20] began to prove that semantic hashing can improve the efficiency of solving some problems without losing accuracy. Subsequently, various hashing methods [21-23] based on deep learning have been developed to improve the usability of semantic hashing.

Simply using the Hamming distance of the binary hash code to calculate the similarity of large texts one by one will cause the same similarity problem. Because a Hamming distance of hundreds of bits can represent only a few hundred similarities, there will be tens of thousands or even hundreds of thousands of similar similarities in tens of millions of data, which will cause many problems. It is a good solution for attaching a common real value weight to each dimension of the hash code [9]. In this solution, only a small amount of weight data needs to be attached to solve the problem of the same similarity, and when the hash table is used, the number of distance calculations and overall time consumed can be reduced [11].

3. Models.

Given a large collection of Q text passages such as Wikipedia, the top k passages relevant to input query are retrieved for the reader component. The reader component further extracts answers from the top k passages. The reason that the retrieval component is needed is that $|Q|$ is much larger than k. For example, in our Wikipedia experiment, $|Q|$ was 21 million, and k was often 20-1000. If the reader directly extracts the answer from a passage in a large collection, the amount of time required will be unacceptable. Our retriever is extended by BPR [6], which is retrieval based on binarized DPR [4]. We firstly introduce BPR and then explain the proposed model.

3.1. Preliminaries.

The binary passage retriever (BPR) [6] was the first retriever to utilize hashing for passage retrieval. By using binary code, BPR reduces the memory cost with little loss of accuracy compared to DPR. BPR uses two independent BERT [3] encoders to encode question q and passage p into continuous embeddings of dimension d:

$$\tilde{x}_q = BERT_q(q), \quad \tilde{x}_p = BERT_p(p) \tag{1}$$

where $\tilde{x}_q \in R^n$ and $\tilde{x}_p \in R^n$. d will change with the encoder; when using the uncased BERT-base, d is 768. The hashing layer computes its binary code from continuous embeddings. To obtain a smaller amount of hash code, continuous embeddings are often reduced to smaller dimensions first. After dimensional reduction, it will eventually become: $x_q \in R^n$ and $x_p \in R^n$. The simple method of obtaining binary code is by using the $sign(\bullet)$ function; however, the back-propagation cannot pass through the sign function, because the gradient of the function is zero.

For training, the BPR uses the scaled tanh function, which approximates the sign function. n is the bit number of the hash code and is often smaller than the embedding dimension. β is a scaling parameter from HashNet [24]. When β increases, the function becomes non-smooth, and the scaled tanh function converges to the sign function when $\beta \to \infty$. The BPR follows the scaling parameter increase strategy from HashNet. In the training phase, the value of β gradually increases from 1, so the values of z_q and z_p are also continuous. However, in the inferencing phase, directly replacing the tanh function with the sign function easily yields the binary hash code.

3.2. Proposed model: WBPR.

The proposed weighted binary passage retriever (WBPR) architecture appears in Figure 1. After passing the BERT encoder, a d-dimensional continuous vector will be obtained, and the WBPR architecture uses a hashing layer to convert a real-valued vector into binary code of any dimension desired. In the training phase, the hash codes of query z_q and passage z_p are also continuous vectors, because they pass the tanh function instead of the sign function. The distance between binary codes is

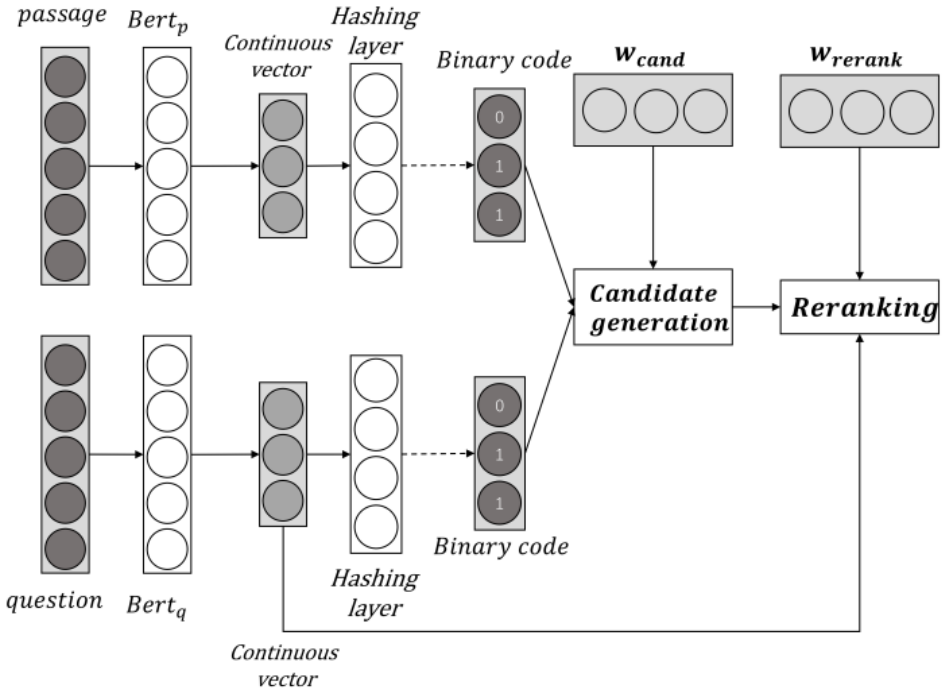

FIGURE 1. Weighted binary passage retriever

generally calculated using the Hamming distance. If we add dimension weight $W \in R^d$ to each dimension, the weighted distance can be represented as follows:

$$Dist\left(z_q, z_p, W\right) = \frac{1}{\sum_{i=1}^{n} W_i} \sum_{i=1}^{n} W_i \left(z_q^i \oplus z_p^i\right) \tag{2}$$

where W_i is the weight of the ith dimension. In training, z_q and z_p are continuous vectors; therefore, $\oplus$ cannot be used. Instead, the inner product is employed in the loss function because the Hamming distance and inner product can be used interchangeably for binary codes.

Similar to the BPR, the proposed architecture consists of *candidate generation* and *reranking*, as shown in Figure 1. In the candidate generation stage, the proposed method uses the hash codes of the question and passage to calculate the weighted Hamming distance after obtaining the hash code using the hash layer. The dimension weight used at this stage is denoted as W_{cand}. The weighted Hamming distance of hash code is $Dist(z_q, z_p, W_{cand})$. In the *reranking* stage, the proposed model uses the hash code of the passage and continuous vector of the question to calculate the weighted distance. The dimension weight used at this stage is denoted as W_{rerank}. The weighted Hamming distance in this stage is $Dist(z_q, z_p, W_{rerank})$.

3.3. **Objective function.** In the *candidate generation* phase, the same ranking loss as in the BPR is utilized as the objective. The difference is that the distance calculated is changed to the weighed Hamming distance. In the training process, z_q and z_p are not real binary codes, but rather approximate hash codes that are calculated by the $\tanh(\cdot)$ function. The loss function in the candidate generation step of the proposed model L_{cand} is as follows:

$$L_{cand} = \sum_{i}^{m} \sum_{j}^{h} \max\left(0, -, Dist\left(z_{q_i}, z_{p_i}^{+}, W_{cand}\right) + Dist\left(z_{q_i}, z_{p_i}^{-}, W_{cand}\right)\right) \tag{3}$$

where W_{cand} is the weight of the candidate generation step, z_p^+ is a positive passage that provides answers to the question, and z_p^- is a negative passage that is irrelevant to the question. In the *reranking* step, like in the DPR, the model is trained by minimizing the negative log-likelihood of the positive passage from the training dataset.

$$L_{rerank} = -\log \frac{\exp\left(Dist\left(x_{q_i}, z_{p_i}^+, W_{rerank}\right)\right)}{\exp\left(Dist\left(x_{q_i}, z_{p_i}^+, W_{rerank}\right)\right) + \sum_{j=1}^{h} \exp\left(Dist\left(x_{q_i}, z_{p_{ij}}^-, W_{rerank}\right)\right)} \quad (4)$$

where W_{rerank} is the weight of the reranking step. In reranking, the distance is calculated based on the continuous vector of the question and hash code of the passage. Thus, in this stage, the effect of W_{rerank} is not as great as that of W_{cand}.

In previous studies, the training dataset was fixed [6], minimizing loss but introducing new types during passage retrieval, termed "high-ranked negative" passages. Including these passages as $z_{p_{ij}}^-$ in Equations (3) and (4) enhances the dynamic learning process of the encoder. Formerly, passage encoder changes during training necessitated comprehensive computations of passage representations, which was time-consuming. However, a recent optimization method [25] demonstrated improved accuracy on a continuous vector base model, despite less effective results in BPR due to its binary code constraint.

In our method, dimension weights are independent of question and passage representations, eliminating the need for constant recalculation of all passage representations. Consequently, both positive and negative passages can change dynamically during training, improving overall results. In practice, the top l results are extracted and reranked to obtain the top k results. From these, the positive passage with the lowest ranking is selected as p_i^+. If no positive passages are present, a positive example is sourced from historical top results. Given cost constraints, negative example sampling requires strategic extraction from all top results. As l is usually greater than k, h passages are randomly selected from the top k results.

4. **Experiment.** The QA experiment performed in this study followed the retriever-reader framework [26], where the retriever selected a small passage list from a large dataset and the reader examined the retrieved passages to obtain the correct answer. The large passage collection Q was the same as the Wikipedia dataset utilized in the DPR. To obtain this dataset, pure text data were extracted from the Wikipedia dump. After preprocessing, 21 million passages remained. The passage index was built using Faiss [27]. As the current Faiss implementation did not offer the weighted Hamming distance, we modified the source code and added the function of the weighted Hamming distance search. Owing to various optimization reasons, the search speed of the version compiled with the source code was much slower than that of the official package, so the weighted search speed that we added the function also caused a certain loss. For a fairer compassion, we conducted experiments using the following two question and answer datasets, Natural questions (NQ) [28] and Trivia QA (TQA) [29]. We used the same BERT-based encoder used by the DPR and BPR and ran the experiments on servers with Intel Xeon(R) Gold 6230 CPUs and four Nvidia Titan RTX GPUs.

4.1. **Main results.** Table 1 presents the top k recall, index size, and query time of the baselines and our proposed methods. Compared with the BPR, which only achieves the recall of the DPR when k is greater than 20, the WBPR performs almost the same as the DPR from top 1 to top 100 recall. Like the BPR, the WBPR reduces the index size from 65 GB to 2 GB. As the dimension weights consist of up to thousands of floats, the size increase of the index is less than 0.01 kB. WBPR-R only differs from the WBPR in the utilization of the *rerank* weight. As the candidate weight is not used, this change only adds a matrix multiplication calculation when reranking among the l results. Thus, the time consumption is almost the same, but the recall is obviously improved.

TABLE 1. Top k recall on NQ and TQA datasets

Methods	Top 1		Top 20		Top 100		Index size	Query time
	NQ	TQA	NQ	TQA	NQ	TQA		
DPR	46.0	53.5	78.4	79.4	85.4	85.0	64.6 GB	456.9 ms
BPR (linear scan; $l = 1000$)	41.1	49.7	77.9	77.9	85.7	84.5	2.0 GB	85.3 ms
BPR (hash table lookup; $l = 1000$)	–		–		–		2.2 GB	38.1 ms
WBPR (linear scan; $l = 1000$)	47.5	50.7	78.4	77.6	85.9	84.0	2.0 GB	115.2 ms
WBPR (hash table lookup; $l = 1000$)	–		–		–		2.2 GB	95.3 ms
WBPR-R (linear scan; $l = 1000$)	41.5	49.0	78.3	76.3	86.0	83.1	2.0 GB	89.2 ms
WBPR-R (hash table lookup; $l = 1000$)	–		–		–		2.2 GB	41.2 ms

4.2. **Effect of number of bits.** In a memory-constrained environment, the index size must be further reduced. For this purpose, a projection layer can be added to the hash layer from Figure 1, which can be done by reducing the hash layer number. Table 2 shows the top k recall results for the NQ dataset with different numbers of bits. In the experiment in which the number of hash codes was reduced, the recall improvement between the WBPR and BPR was further increased. In the experiment with 128 bits, the recall of the WBPR was significantly better than that of the BPR. Table 2 shows the recall difference between the WBPR and BPR.

TABLE 2. Top k recall on NQ dataset with different numbers of bits

Number of bits	Methods	Top 1	Top 5	Top 20	Top 50	Top 100	Index size
768 bits	BPR	41.10	66.23	77.90	82.80	85.70	2.0 GB
	WBPR	**47.45**	**66.93**	**78.37**	**83.10**	**85.87**	
512 bits	BPR	22.80	45.87	62.44	70.78	75.54	1.3 GB
	WBPR	**23.52**	**46.26**	**62.85**	**71.47**	**76.29**	
256 bits	BPR	17.37	38.42	56.07	65.84	71.63	0.67 GB
	WBPR	**18.25**	**40.00**	**57.76**	**67.29**	**72.27**	
128 bits	BPR	11.58	29.25	47.20	59.09	66.73	0.42 GB
	WBPR	**13.55**	**32.66**	**51.58**	**62.22**	**67.87**	

As the subsequent reader extracts the correct results from the retrieval results, the slight improvement in the recall of the retriever when the recall is very high is completely different from that when it is very low. When the recall is very low, reader error is often caused by the absence of an answer to extract; thus, slight recall improvement often directly leads to an improvement in the accuracy of the final answer. However, when the recall is already high, the reader error is often not due to the absence of an answer in the candidate. Therefore, recall improvement is often not critical in this situation. The index size is limited in the WBPR, making it very meaningful to improve the recall.

4.3. **Effect of candidate number.** Table 3 shows the recall for different candidate numbers. Several observations were made based on these results. First, the top 1 and top 20 recalls of the BPR do not change from $l = 200$ to $l = 2000$; the same is true of the WBPR. This finding also shows that no matter how many candidates are extracted, the positive passage of some questions can be ranked first as long as they pass the appropriate

TABLE 3. Top 1, top 20 and top 100 recall with different numbers of candidates

	Top 1		**Top 20**		**Top 100**	
	BPR	WBPR	BPR	WBPR	BPR	WBPR
$l = 200$	41.10	47.52	77.90	78.34	85.40 (-0.3)	85.87
$l = 500$	41.10	47.52	77.90	78.34	85.60 (-0.1)	85.87
$l = 1000$	41.10	47.52	77.90	78.34	85.70	85.87
$l = 2000$	41.10	47.52	77.90	78.34	85.70	85.87

reranking. Second, the top 100 recall of the BPR decreases when $l = 500$ to $l = 200$. Based on reference to Table 3, this decrease is most likely caused by the same distance issue. As the dimension weight solves this issue, the recall does not change when l is greater than 1000.

5. **Conclusions.** This paper presented a novel weighted semantic hashing method that uses dimension weights to improve passage retrieval performance. This method can learn the hash code in a weighted Hamming space and is more effective than previous hashing methods. In addition, the proposed top k negative sampling technique can make training more effective when hashing is weighed. Benefitting from the weighted hashing and training technique, the proposed approach not only outperforms the previous hashing method in terms of recall, but also is comparable to the dense retrieval method that has greater memory requirements. In other words, compared with the previous hashing method that loses part of the recall, our method reduces the memory by 30 times without losing even a small amount of recall.

The proposed WBPR cannot surpass the BPR model in terms of query time, but in theory, the weighted method has a faster lookup algorithm than the non-weighted hash table lookup. However, we have not found a better implementation method for the existing Faiss. If we could fully utilize the acceleration mechanism of Faiss, we could use the acceleration algorithm of weighted hashing to achieve a breakthrough in query time.

Acknowledgment. This work was supported in part by the National Key R&D Program of China (No. 20227D0116306), and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022H1D8A3037394).

REFERENCES

[1] L. Ma, M. Li, W.-N. Zhang, J. Li and T. Liu, Unstructured text enhanced open-domain dialogue system: A systematic survey, *ACM Transactions on Information Systems*, vol.40, no.1, pp.1-44, 2022.

[2] D. Chen, A. Fisch, J. Weston and A. Bordes, Reading Wikipedia to answer open-domain questions, *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, pp.1870-1879, 2017.

[3] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp.4171-4186, 2019.

[4] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen and W. T. Yih, Dense passage retrieval for open-domain question answering, *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.6769-6781, 2020.

[5] K. Lee, M. W. Chang and K. Toutanova, Latent retrieval for weakly supervised open domain question answering, *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics*, pp.6086-6096, 2019.

[6] I. Yamada, A. Asai and H. Hajishirzi, Efficient passage retrieval with hashing for open-domain question answering, *arXiv.org*, arXiv: 2106.00882, 2021.

[7] R. Salakhutdinov and G. Hinton, Semantic hashing, *International Journal of Approximate Reasoning*, vol.50, no.7, pp.969-978, 2009.
[8] J. M. Johnson and T. M. Khoshgoftaar, Survey on deep learning with class imbalance, *Journal of Big Data*, vol.6, no.1, pp.1-54, 2019.
[9] Q. Wang, D. Zhang and L. Si, Weighted hashing for fast large scale similarity search, *Proc. of the 22nd ACM International Conference on Information & Knowledge Management*, pp.1185-1188, 2013.
[10] Y. Cao, J. Liu, H. Qi, J. Gui, K. Li, J. Ye and C. Liu, Scalable distributed hashing for approximate nearest neighbor search, *IEEE Transactions on Image Processing*, vol.31, pp.472-484, 2021.
[11] J. Gui, Y. Cao, H. Qi, K. Li, J. Ye, C. Liu and X. Xu, Fast kNN search in weighted hamming space with multiple tables, *IEEE Transactions on Image Processing*, vol.30, pp.3985-3994, 2021.
[12] O. Khattab, C. Potts and M. Zaharia, Relevance-guided supervision for OpenQA with ColBERT, *Transactions of the Association for Computational Linguistics*, vol.9, pp.929-944, 2021.
[13] C. D. Manning, P. Raghavan and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, Cambridge, 2008.
[14] S. Robertson and H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, *Foundations and Trends in Information Retrieval*, vol.3, no.4, pp.333-389, 2009.
[15] T. Wolfson, M. Geva, A. Gupta, M. Gardner et al., Break it down: A question understanding benchmark, *Transactions of the Association for Computational Linguistics*, vol.8, pp.183-198, 2020.
[16] S. Min, D. Chen, L. Zettlemoyer and H. Hajishirzi, Knowledge guided text retrieval and reading for open domain question answering, *arXiv.org*, arXiv: 1911.03868, 2019.
[17] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, Indexing by latent semantic analysis, *Journal of the American Society for Information Science*, vol.41, no.6, pp.391-407, 1990.
[18] D. Antons, C. F. Breidbach, A. M. Joshi and T. O. Salge, Computational literature reviews: Method, algorithms, and roadmap, *Organizational Research Methods*, vol.26, no.1, pp.107-138, 2023.
[19] P. Lewis, E. Perez, A. Piktus, F. Petroni et al., Retrieval-augmented generation for knowledge-intensive NLP tasks, *Advances in Neural Information Processing Systems*, vol.33, pp.9459-9474, 2020.
[20] S. Chaidaroon and Y. Fang, Variational deep semantic hashing for text documents, *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.75-84, 2017.
[21] D. Shen, Q. Su, P. Chapfuwa, W. Wang et al., Nash: Toward end-to-end neural architecture for generative semantic hashing, *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics*, pp.2041-2050, 2018.
[22] R. Xuan, J. Shim and S. G. Lee, Deep semantic hashing using pairwise labels, *IEEE Access*, vol.9, pp.91934-91949, 2021.
[23] Y. Zhang and H. Zhu, Doc2hash: Learning discrete latent variables for documents retrieval, *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp.2235-2240, 2019.
[24] Z. Cao, M. Long, J. Wang and P. S. Yu, HashNet: Deep learning to hash by continuation, *Proc. of the IEEE International Conference on Computer Vision*, pp.5608-5617, 2017.
[25] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang and S. Ma, Optimizing dense retrieval model training with hard negatives, *arXiv.org*, arXiv: 2104.08051, 2021.
[26] R. Das, S. Dhuliawala, M. Zaheer and A. McCallum, Multi-step retriever-reader interaction for scalable open-domain question answering, *International Conference on Learning Representations*, 2018.
[27] J. Johnson, M. Douze and H. Jégou, Billion-scale similarity search with GPUs, *IEEE Transactions on Big Data*, vol.7, no.3, pp.535-547, 2019.
[28] T. Kwiatkowski, J. Palomaki, O. Redfield et al., Natural questions: A benchmark for question answering research, *Transactions of the Association for Computational Linguistics*, vol.7, pp.453-466, 2019.
[29] M. Joshi, E. Choi, D. S. Weld and L. Zettlemoyer, TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension, *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, pp.1601-1611, 2017.