Topic-Grained Text Representation-based Model for Document Retrieval

*Mengxue Du, *Shasha Li, Jie Yu✉, Jun Ma✉, Bin Ji, Huijun Liu, Wuhang Lin and Zibo Yi

College of Computer, National University of Defense Technology, Changsha, Hunan Province, China
{dumengxuenudt,shashali,yj,majun,
jibin,liuhuijun,wuhanglin,yizibo14}@nudt.edu.cn

Abstract. Document retrieval enables users to find their required documents accurately and quickly. To satisfy the requirement of retrieval efficiency, prevalent deep neural methods adopt a representation-based matching paradigm, which saves online matching time by pre-storing document representations offline. However, the above paradigm consumes vast local storage space, especially when storing the document as word-grained representations. To tackle this, we present **TGTR**, a **T**opic-**G**rained **T**ext **R**epresentation-based **Model** for document retrieval. Following the representation-based matching paradigm, TGTR stores the document representations offline to ensure retrieval efficiency, whereas it significantly reduces the storage requirements by using novel topic-grained representations rather than traditional word-grained. Experimental results demonstrate that compared to word-grained baselines, TGTR is consistently competitive with them on TREC CAR and MS MARCO in terms of retrieval accuracy, but it requires less than 1/10 of the storage space required by them. Moreover, TGTR overwhelmingly surpasses global-grained baselines in terms of retrieval accuracy.

Keywords: Neural Retrieval · Text Representation · Topic Granularity · Space Compression

1 Introduction

Recently, deep learning based semantic representations have attracted much research attention and been widely used in the document retrieval field. Recent methods propose to fine-tune deep pre-trained language models (PLMs) such as BERT [3] to assess matching degrees of query-document pairs [10, 25, 3]. They achieve the state-of-the-art performance of the document retrieval task by concatenating query-document pair and feeding it into a PLM to calculate the matching degree. Unfortunately, despite these methods achieve great success, they come at a steep increase in time cost, which is unacceptable in practical application scenarios.

In order to improve the retrieval speed, researchers propose a representation-based framework, where they encode query and document into word-grained

representations [10], as shown in Fig. 1(a). And then they assess the matching degree of a query and a document pair by calculating the similarity of their representations. Benefit from the decoupling computation of queries and documents, the representation-based framework can pre-store document representations offline. Thus the online retrieval only needs to encode the query while it obtains the document representations from local storage directly. However, the representation-based framework come at a steep increase in space cost to store document representations. For example, when using ColBERT [10] to generate document representations, it requires 154 GiBs to store the TREC CAR corpus and 632 GiBs to store the MS MARCO corpus, where the document sizes of two corpora are only 2.9 GiBs and 15.6 GiBs, respectively. In this paper, we explore a novel method to compress document representations, with the goal of saving storage space and guaranteeing the retrieval accuracy as well.
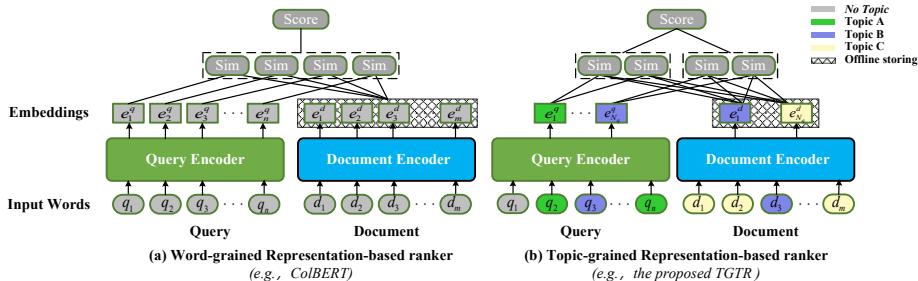

Fig. 1. The matching paradigms of word-grained retriever (a) and the proposed method (b). Given a query and a document, (a) and (b) encode them into word-level embeddings and topic-level embeddings, respectively. (b) reduce the length of document's representation from the word count level to the topic count level, which means it significantly compresses the space cost of offline storing documents' representations.

To address the above issues, we proposes TGTR, a topic-grained text representation-based document retrieval model, as shown in Fig. 1(b). To be specific, we first model the topics distribution of documents and queries to obtain every word's latent topics, and then use Attention network to obtain topic-level embeddings by fusing words' contextual embeddings with the same topic. The motivation is drawn from the fact that, in general, users are only interested in documents consisting of information closely related to their search topic. The information in a document may cover multiple topics, and users tend to pay more attention to those parts of the document which are closely related to the query topic and less attention to the remainder. Unfortunately, previous document retrieval models [9, 7, 10, 12, 19] ignore to take the topic information into account. In addition, the problem is particularly acute for long documents.

We see the following intuitive benefits when using topic-grained representations of queries and documents to retrieval documents.

1) Compress the space cost of storing document representations. Compared to word-grained retrievers, the proposed topic-grained retriever can compress

the size of the document representations by one order-of-magnitude, and the compression rate increases as the document length increases.

2) Keep a balance of the amount of information between each embedding in query and document's representations. We think it's the main reason why TGTR can achieve better retrieval accuray than the alternative methods described in Section 2.

3) Break the existing information fusion that follows the structure of the article. We fuse the contents sharing the same topic together across the whole article. It's a process of distilling the representive information of a document. Furthermore, we find that no-topic words are frequently filler words. Which indicates our model is effective at filtering out redundant information.

In summary, this work makes the following contributions:

1) We propose a novel document retrieval model that introduces topic-grained representation to the task for the first time;

2) Our model guarantees retrieval accuracy while significantly compressing the storage space of the document representations;

3) Our model obtains competitive performance compared to all baselines on two benchmark datasets in terms of retrieval accuracy, but it requires less than 1/10 of the space cost compared to them.

2   Related Work

Classical information retrieval (IR) systems rely on exact lexical match [21], we call them lexical retrievers. Lexical retrievers can process queries very quickly. Nowadays, they are still widely used in production systems [7]. Recently, researchers have utilized deep learning to improve traditional lexical retrievers, including document expansions [17, 16], query expansions [13] and term weight estimation [2].

In the past few years, information retrieval researchers have introduced a range of neural models for semantic retrieval [25, 7, 10, 6, 9, 8, 12, 19]. Due to the specific requirements of time efficiency, researchers proposed the representation-based retrieval framework. [3, 6, 7, 9, 10, 11, 19, 23].

Khattab. Omar et al. [10] first proposed a word-grained representation-based retriever, we call this type of models word-grained retrievers. Word-grained retrievers provides state-of-the-art performance at that time while resulting in significant storage overhead. COIL [7] is another word-grained retriever which stores the token embedding in an inverted list. Representing queries and/or documents separately with a single embedding is an important method to compress document representations, which we call global-grained retrievers [6, 9, 19]

Global-grained retrievers also generate word-level embeddings firstly, but then they fuse the sequence of embeddings into one by various means. Sentence-BERT [20] explores the effect of using 1) [CLS] embedding; 2) average pooling; 3) max pooling to fuse the BERT embedding sequence, respectively. However, This type of models can seriously impair retrieval accuracy. we attribute it to

that the amount of information between query and document in the real world is often asymmetric ($|query| \ll |document|$), which leads to an imbalance of the amount of information between each embedding in query and document's representations.

In summary, current representation-based retrievers face the tradeoff of the space cost (document representations) and retrieval accuracy. TGTR effectively reduces the cost of space by constructing the topic-grained representation, without compromising retrieval accuracy.

3   TGTR

In this section, we present our topic-grained text representation-based model for document retrieval. Before we present the framework, some preliminary about representation-based matching paradigm are introduced. Then the TGTR framework are described in detail.

3.1   Preliminary

In the field of document retrieving, specially for deep models, it's very common to assess the matching degree of a query-document pair by representing the query and/or document as a sequence of vectors which we called representation-based matching paradigm. Given a query sequence $Q = [q_1, q_2, ..., q_n]$ and a document sequence $D = [d_1, d_2, ..., d_m]$, both q_i and d_j represent a word. Firstly, encoding a query and document into representations E^q and E^d, then calculating the similarity between E^q and E^d [22]. As Fig. 1(a) shows, traditional word-grained retrievers encode every word into a fixed-length embedding $E^q = [e_1^q, e_2^q, ..., e_n^q]$ and $E^d = [e_1^d, e_2^d, ..., e_m^d]$.

By design, the representation-based matching paradigm isolates almost all of the computations between queries and documents to enable pre-computing document representations offline [10]. It proceeds over the documents in the collection in batches, once the documents' representations are produced, they are saved to disk using 32-bit or 16-bit values to represent each dimension. In Fig. 1, we use the rectangular box with decorative pattern to identify the part of offline storing.

Generally, per embedding in representation is about hundreds of dimensions and storing a dimension needs at least 16-bit. The number of a document's embeddings stored by word-grained retrievers is approximately equal to the document length, making huge space cost. As Fig. 1(b) shows, we propose to encode the document with N_d representative topics into topic-grained representations $E^d = [e_1^d, e_2^d, ..., e_{N_d}^d]$ rather the traditional word-grained representations. The idea's purpose is to reduce the number of the document's embeddings to compress the space cost of storing documents' representations.

3.2   Model Architecture

Fig. 2 depicts the architecture of TGTR, which comprises four components. We will cover these components in detail in this section.
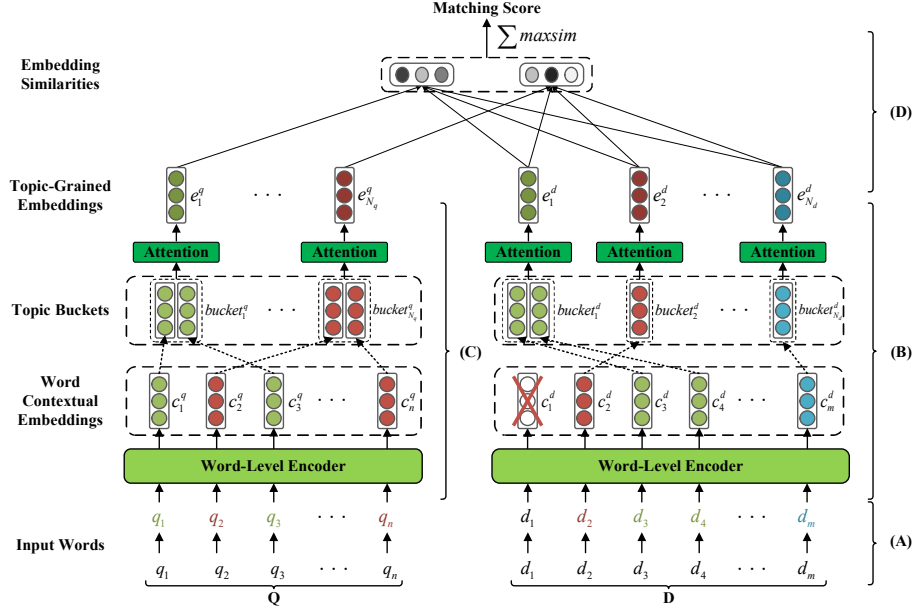

Fig. 2. The architecture of TGTR. (A) Topic Recognizer; (B) Query Encoder; (C) Document Encoder; and (D) Matching Assessment Mechanism. Given an input query sentence Q and a document sentence D, (A) recognizes latent topics of every word in documents and queries, and then (B) and (C) encode the query and the document to sequences of topic-grained embeddings by three stages, separately. Finally, (D) uses maximum similarity operation to output final matching score between the query and the document.

Topic Recognizer   The topic recognizer uses traditional topic generation model to recognize latent topics of every word in documents and queries. As Fig. 2(A) shows, given a query sequence $Q = [q_1, q_2, ..., q_n]$ and a document sequence $D = [d_1, d_2, ..., d_m]$, topic recognizer gives their words different topic colors by analyzing their topics distributions.

To be specific, we use Latent Dirichlet Allocation (LDA) [1] to model documents and queries' latent topics. Algorithm 1 depicts the process of obtaining words' topics. Firstly, we obtain the text-topic distribution Array as well as the topic-word distribution Array (line 1-2). We then set threshold θ_t to extract the representative topics for each text (line 3-8). In the same manner, we set fixed threshold θ_{wf} and ratio threshold θ_{wr} to extract the representative words in a text under its representative topics (line 9-11). Finally, a two-dimensional

table M is generated for each text (line 12). The dim of row in M represents a word, while the dim of column represents a potential topic. Note a word may have more than one or zero latent topics. The words without any latent topics are considered meaningless and discarded after helping other words to generate contextual embeddings (see the red cross symbol in Fig. 2).

Algorithm 1 Topic Recognizer of TGTR

Input:
    The text-words array, text2words;
    The number of texts, m;
    The number of topics, k;
    The trained topic model, LDA;
    The word frequency builder, Vectorizer;
    The threshold for extracting representative topics , θ_t;
    The fixed and ratio threshold for extracting representative words, θ_{wf}, θ_{wr};
Output:
    The list of potential topics of words in all texts, text2word2topics;

1: textVectorizer $\leftarrow$ Vectorizer.transform(text2words)
2: text2topics, topic2words $\leftarrow$ LDA.transform(textVectorizer)
3: **for** $d \in [1, 2, ..., m]$ **do**
4:     $M \leftarrow [\,]$, text2word2topics $\leftarrow [\,]$
5:     **for** $t \in [1, 2, ..., k]$ **do**
6:         **if** text2topics$[d, t] \geq \theta_t$ **then**
7:             Get the distribution of words in the d-th text
8:                 under the t-th topic dis$_{dt}$ $\leftarrow$ topic2words
9:             $l \leftarrow |\text{dis}_{dt}|$
10:             **for** $w \in [1, 2, ..., l]$ **do**
11:                 **if** dis$_{dt}[w] \geq \theta_{wf}$ **or** dis$_{dt}$.getorder$(w)/l \leq \theta_{wr}$ **then**
12:                     $M[w]$.append(t)
13:     text2word2topics.append(M)
14: **return** text2word2topics

Document Encoder   We then encode the document to a sequence of fixed-length embeddings. This part comprises three stages.

Stage 1: Encode word-level contextual embeddings   Given a document D, TGTR first maps each word d_i into its contextual embedding c_i^d by using Word-Level Encoder (WLE). Though we can complete this part of work by using methods such as in [5, 3], We focus on BERT [3] to keep consistent with the major baseline. Note BERT uses WordPiece embeddings with a 30,000 token vocabulary, thus a word can be tokenized to several tokens. Strictly speaking, the i-th word's contextual embedding c_i^d may comprises more than one token embedding, which we hope readers will notice. The process of this stage is summarized as Equation 1.

$$[c_1^d, c_2^d, ..., c_m^d] := \text{WLE}(d_1, d_2, ..., d_m) \tag{1}$$

Stage 2: Word-topic mapping   As mentioned above, we obtain a two-dimensional word-topics table M for every document by modeling latent topics. The dim of

row in M represents a word, while the dim of column represents a potential topic. Every word's contextual embedding obtained in Stage 1 is mapped to the buckets corresponding to topics they have. A word may be mapped into multiple buckets or filtered out (regarded as meaningless). The bucket corresponding to the i-th topic is marked as $bucket_i^d$. The process of this stage is summarized as Equation 2, where N_d is the number of representative topics the document d has.

$$[bucket_1^d, bucket_2^d, ..., bucket_{N_d}^d] := \text{Mapping}(c_1^d, c_2^d, ..., c_m^d) \qquad (2)$$

Stage 3: Generate topic-grained representation   The model TGTR uses **Attention** network to obtain topic-level embeddings, which we call topic-grained representation. Considering different words with the same topic have different amount of information, we assign different weights to different words. For the bucket corresponding to the t-th topic $bucket_t^d : [u_1, u_2, ..., u_{B_t}]$ outputed by stage 2, denote the attention weight of u_i as α_i:

$$\alpha_i = q_t{}^T \tanh(W \times u_i + b) \qquad (3)$$

$$\alpha_i = \frac{\exp(\alpha_i)}{\sum_{j=1}^{B_t} \exp(\alpha_j)} \qquad (4)$$

where W and b are parameters, q_t is the attention query vector, $tanh$ is the activation function and B_t is the size of the t-th topic bucket. The final embedding of t-th topic e_t^d is the summation of the word-level embeddings in $bucket_t^d$ weighted by their attentions.

$$e_t^d = \sum_{i=1}^{B_t} \alpha_i u_i \qquad (5)$$

Query Encoder   Our query encoder has a very similar architecture with document encoder, they share model parameters but have a few difference in input processing. We prepend BERT's start token [CLS] followed by a special token [D] when input a document sequence. In the same manner, we prepend BERT's start token [CLS] followed by a special token [Q] when input a query sequence.

Matching Assessment Mechanism   Finally, we use a maximum similarity (MaxSim) operation to output our final matching score. Given the query's topic-grained representation $E^q : [e_1^q, e_2^q, ..., e_{N_q}^q]$ and the document's topic-grained representation $E^d : [e_1^d, e_2^d, ..., e_{N_d}^d]$, the matching score of query q and document d is assessed by MaxSim operation between E^q and E^d. To be specific, we applies MaxSim between one of the query embeddings and all of the document's embeddings, then we sum all items up as final score $S(Q, D)$. The process of this part is summarized as Equation 6.

$$S(Q, D) = \sum_{i \in [|E^q|]} \max_{j \in [|E^d|]} e_i^q (e_j^d)^T \qquad (6)$$

Notice our Matching Assessment Mechanism has no trainable parameters.

Training   The training objective is to learn representations of queries and documents so that query-positive document pairs have higher matching score than the query-negative documents pairs in training data. Given a query Q together with its positive documents D^+ and m negative documents. $\{D_i^-\}_{i=1}^m$, we minimize the loss function:

$$L(Q, D^+, \{D_i^-\}_{i=1}^m) = -\log \frac{\exp(S(Q, D^+))}{\exp(S(Q, D^+)) + \sum_{i=1}^m \exp(S(Q, D_i^-))} \quad (7)$$

4   Experiment Methodology

4.1   Datasets

Following previous work [10], our experiments use two datasets, which differ in data size, to evaluate our model in document retrieving tasks.
TREC CAR. TREC CAR is introduced by Dietz et al. [4] in 2017, is a composite data set based on Wikipedia containing approximately 29 million articles. Our assessment was performed on the test set used in TREC 2017 CAR, which contained 2,254 queries.
MS MARCO. MS MARCO [15] is a dataset introduced by Microso in 2016 for reading comprehension and adapted in 2018 for retrieval. It is a collection of 8.8M passages from Web pages, which were gathered from Bing's results to 1M real-world queries.

4.2   Baseline Methods

We adopt three types of baselines for comparison.
Lexical Retriever. Lexical Retriever retrieve document based on lexical matching rather than semantic matching. In this type, we choose three traditional methods [24, 21, 14] and three network methods [17, 16, 2] as our baselines.
Global-grained Retriever. Global-grained retriever retrieve document with global-grained representations of queries and documents. In this type, we choose BERT [3] and DPR [9] as our baselines.
Word-grained Retriever. Global-grained retriever retrieve document with word-grained representations of queries and documents. In this type, we choose ColBERT [10] and COIL [7] as our baselines.

5   Experiment Details

5.1   Implementation Details

The complete training details are given below:

– We fit LDA model by using Scikit-learn machine learning library [18]. We apply variational inference with expectation-maximization to learn model's parameters and get the distributions described in Section 3. The number of latent topics K is a hyper-parameters here, and we set other two hyper-parameters α and η to $1/K$ by default.

- We choose the max query length as 32 and the max doc length as 180 at dataset MS MARCO. Since TREC is much larger than MS MARCO, we set max query length 48 and max doc length 250 in TREC.

- We use BERT as pre-trained word-level embedding encoder to embed the query and document sentences with the embedding dimension of 768 and the vocab size of 30522.

- We then apply attention operation for every topic buckets by different query vectors. The parameters W and b are shared by all buckets.

- The dimension of final topic-level embedding dim is 768. We passes the embeddings through a linear layer with no activations to control their dimensions. As we discuss later in more detail, we typically fix dim range as (64, 128, 256, 512, 768). We set dim=256 by default.

5.2   Experiment Results

Table 1 shows the retrieving performance of TGTR and our baselines over two datasets.

Compared to word-grained retrievers The results show that TGTR performs almost 10 times better than the word-grained baselines in terms of space cost with no loss in terms of retrieval accuracy on MS MARCO. On TREC CAR, TGTR performs almost 12 times better than the word-grained baselines in terms of space cost with 2.3% and 2.0% loss in terms of MRR@10 and MAP on MS MARCO.

Compared to global-grained retrievers The results show that TGTR overwhelmingly outperforms global-grained retrievers in terms of space cost and retrieval accuracy over both datasets. Note our model outperforms global-grained retrievers in terms of space cost because we reduce the embedding dimension by passing the original embeddings through a linear layer.

Compared to lexical retrievers The results show that TGTR overwhelmingly outperforms lexical retrievers in terms of retrieval accuracy over both datasets. Note lexical retrievers don't need store documents' representations, so the compare between our model and them in terms of space cost is not available.

Summary. Compared to word-grained baselines, TGTR is consistently outperforming them on MS MARCO and be competitive with them on TREC CAR in terms of retrieval accuracy, but it performs almost 10 times better than them in terms of space cost. Moreover, TGTR overwhelmingly outperforms lexical and global-grained baselines in terms of retrieval accuracy.

Table 1. Retrieving performances of TGTR and baseline models. We report the performances of our model with the embedding dimension dim=256. Improvement, degradation or equivalent with respect to TGTR in terms of MRR@10, Recall@1K and MAP is indicated $(+/-/-)$. The unit of 'Space' is (GiBs). Results not applicable are denoted 'n.a.'.

(a) Performance Comparisons on MS MARCO.

Method	Space(GiBs)		MRR@10		Recall@1K	
Lexical Retriever						
BM25	n.a.	n.a.	0.187	−48.2%	0.857	−11.5%
Doc2query	n.a.	n.a.	0.215	−40.4%	0.891	−8.0%
DeepCT	n.a.	n.a.	0.243	−32.7%	0.910	−6.0%
DocTTTTTquery	n.a.	n.a.	0.277	−23.3%	0.947	−2.2%
Global-grained Retriever						
BERT	25.3	×1.6	0.310	−14.1%	0.929	−4.0%
DPR	n.a.	n.a.	0.311	−13.9%	0.952	−1.7%
Word-grained Retriever						
COIL	n.a.	n.a.	0.355	−1.7%	0.963	−0.5%
ColBERT	154.0	×9.9	0.360	-0.3%	**0.968**	-
Topic-grained Retriever						
TGTR	**15.6**	**×1**	**0.361**	-	**0.968**	-

(b) Performance Comparisons on TREC CAR.

Method	Space(GiBs)		MRR@10		MAP	
Lexical Retriever						
BM25	n.a.	n.a.	n.a.	n.a.	0.153	−50.2%
TextRank	n.a.	n.a.	0.160	−63.0%	0.120	−60.9%
Doc2query	n.a.	n.a.	n.a.	n.a.	0.181	−41.0%
DeepCT	n.a.	n.a.	0.332	−23.3%	0.246	−19.9%
Global-grained Retriever						
BERT	83.2	×1.6	0.376	−13.2%	0.273	−11.1%
Word-grained Retriever						
ColBERT	632.1	×12.3	**0.443**	**+2.3%**	**0.313**	**+2.0%**
Topic-grained Retriever						
TGTR	**51.4**	**×1.0**	0.433	-	0.307	-

6   Analysis

6.1   A Comparison of Trade-off Quality

In this section, we assess the trade-off quality between space efficiency and retrieval accuracy of three types of representation-based retrievers. We use the quotient of MRR and Space as the trade-off score. We use BERT and ColBERT to represent the global-grained retriever and word-grained retriever, separately. Topic-grained Retriever is our model.

Table 2 shows the results. It seems that our model significantly outperforms other two types of retrievers in term of trade-off quality and the word-grained retriever performs the worst.

Table 2. Comparisons of Trade-off Quality among Three Types of Retrievers.

Method	MRR/Space (1e-3)			
	MS MARCO		**TREC CAR**	
Global-grained Retriever	12.3	−46.8%	4.5	−46.4%
Word-grained Retriever	2.3	−90.0%	0.7	−91.7%
Topic-grained Retriever	**23.1**	-	**8.4**	-

6.2   Embeddings Dimension and Bytes per Dimension

Two of the most attractive features in our model is the embeddings dimension and the bytes per dimension. Fig. 3 shows the impact of above two features on the model performance. As Fig. 3(a) shows, retrieval accuracy increases sublinearly
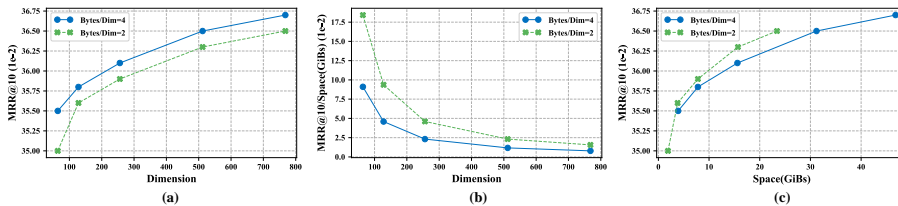

Fig. 3. (a) and (b) show the impact of embeddings dimension and the bytes per dimension on MRR@10 and trade-off quality (MRR/Space), separately. (c) shows MRR@10 vs Space(GiBs) as functions of the embeddings dimension and the bytes per dimension.

with the increase of above two features in our model. Fig. 3(b) clearly shows that it might contribute to higher trade-off quality by reducing the above two features. As Fig. 3(c) shows, retrieval accuracy increases sublinearly with the increase of space cost in our model. It seems that when the embedding dimension is small enough, further compression can cause great accuracy damage.

7   Conclusions

This paper presents TGTR, a novel retrieval model that employs topic-grained text representation for document retrieval. The key of our model is modeling the topics distribution of documents and queries to obtain every word's latent topics, and then using Attention network to obtain topic-level embeddings by fusing words' contextual embeddings with the same topic. Our experiments on MS MARCO and TREC benchmark datasets demonstrates the advantage of representing texts as topic-grained embeddings for document retrieval task. These results suggest that our model guarantees retrieval accuracy while significantly compressing the storage space of the document representations.

References

1. Blei, D.M., Ng, A.Y.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
2. Dai, Z., Callan, J.: Context-aware term weighting for first stage passage retrieval. In: SIGIR. pp. 1533–1536 (2020)
3. Devlin, J., et al.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
4. Dietz, L., Verma, M.: Trec complex answer retrieval overview. In: TREC (2017)
5. Feng, Z., Tang, D., et al.: Pretraining without wordpieces: Learning over a vocabulary of millions of words. arXiv:2202.12142 (2022)
6. Gao, L., Callan, J.: Condenser: a pre-training architecture for dense retrieval. arXiv:2104.08253 (2021)
7. Gao, L., Dai, Z.: Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. arXiv:2104.07186 (2021)
8. Guo, J., Fan, Y.: A deep relevance matching model for ad-hoc retrieval. In: the 25th CIKM. pp. 55–64 (2016)
9. Karpukhin, V., Oğuz, B.: Dense passage retrieval for open-domain question answering. arXiv:2004.04906 (2020)
10. Khattab, O., Zaharia, M.: Colbert: Efficient and effective passage search via contextualized late interaction over bert. In: SIGIR. pp. 39–48 (2020)
11. Lu, S., He, D.: Less is more: Pretrain a strong siamese encoder for dense text retrieval using a weak decoder. In: 2021 EMNLP. pp. 2780–2791 (2021)
12. Ma, X., Guo, J.: Prop: Pre-training with representative words prediction for ad-hoc retrieval. In: the 14th WSDM. pp. 283–291 (2021)
13. Mao, Y., He, P.: Generation-augmented retrieval for open-domain question answering. arXiv:2009.08553 (2020)
14. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: the 2004 EMNLP. pp. 404–411 (2004)
15. Nguyen, T., Rosenberg, M.: Ms marco: A human generated machine reading comprehension dataset. In: CoCo@ NIPS (2016)
16. Nogueira, R., Lin, J., Epistemic, A.: From doc2query to doctttttquery. Online preprint **6** (2019)
17. Nogueira, R., Yang, W., Lin, J., Cho, K.: Document expansion by query prediction. arXiv:1904.08375 (2019)
18. Pedregosa, F., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
19. Qu, Y., Ding, Y.: Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. arXiv:2010.08191 (2020)
20. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv:1908.10084 (2019)
21. Robertson, S.E., et al.: Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In: SIGIR'94. pp. 232–241. Springer (1994)
22. Sun, Q., Wu, Y.: A multi-level attention model for text matching. In: International Conference on Artificial Neural Networks. pp. 142–153. Springer (2018)
23. Zamani, H., et al.: From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In: the 27th CIKM. pp. 497–506 (2018)
24. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to ad hoc information retrieval (2001)
25. Zheng, Z., Hui, K.: Bert-qe: contextualized query expansion for document re-ranking. arXiv:2009.07258 (2020)