SCV: Light and Effective Multi-Vector Retrieval with Sequence Compressive Vectors

Cheoneum Park[1]*, Seohyeong Jeong[2], Minsang Kim[2], Kyeongtae Lim[3] Yonghoon Lee[2]
[1]Hanbat National University
[2]SK Telecom
[3]Seoul National University of Science and Technology
parkce@hanbat.ac.kr, jseoh95@gmail.com
{minsang.0804, yhlee95}@sktelecom.com, ktlim@seoultech.ac.kr

Abstract

Recent advances in language models (LMs) has driven progress in information retrieval (IR), effectively extracting semantically relevant information. However, they face challenges in balancing computational costs with deeper query-document interactions. To tackle this, we present two mechanisms: 1) a light and effective multi-vector retrieval with sequence compression vectors, dubbed SCV and 2) coarse-to-fine vector search. The strengths of SCV stems from its application of span compressive vectors for scoring. By employing a non-linear operation to examine every token in the document, we abstract these into a span-level representation. These vectors effectively reduce the document's dimensional representation, enabling the model to engage comprehensively with tokens across the entire collection of documents, rather than the subset retrieved by Approximate Nearest Neighbor. Therefore, our framework performs a coarse single vector search during the inference stage and conducts a fine-grained multi-vector search end-to-end. This approach effectively reduces the cost required for search. We empirically show that SCV achieves the fastest latency compared to other state-of-the-art models and can obtain competitive performance on both in-domain and out-of-domain benchmark datasets.

1 Introduction

Information retrieval (IR) is the task of finding a set of relevant documents from an indexed collection for a given query (Manning et al., 2008). Recently, in modern Retrieval-Augmented Generation (RAG) models (Shi et al., 2024; Anantha and Vodianik, 2024; Baek et al., 2023; Jeong et al., 2024), an effective neural IR is crucial for sourcing accurate and relevant clues in real-time, significantly improving the quality and contextual appropriateness of generated content. Neural IR can be largely divided into two categories; single-vector retrieval and multi-vector retrieval. The former approach (Karpukhin et al., 2020; Formal et al., 2021) relies on a single vector representation extracted from a document and calculates the relevance score with representations pooled from both queries and documents. In contrast, multi-vector retrieval methods such as ColBERT, GTR, COIL, and CITADEL (Khattab and Zaharia, 2020; Ni et al., 2022; Gao et al., 2021; Li et al., 2023) show promising performance by representing document text as token collections rather than single vectors.

However, Khattab and Zaharia (2020) requires indexing all tokens in a collection of documents, leading to significant memory and computational burdens. To reduce this burden, a multi-stage retrieval approach is adopted. In the first stage, indexing and searching for relevant documents given the query are performed using approximate nearest neighbor (ANN) (Macdonald and Tonellotto, 2021). In the second stage, the top-k results are output by re-ranking, which is trained based on the extracted documents. Gao et al. (2021); Li et al. (2023) have further improved multi-vector retrieval methods by computing the score between the query and the document using semantically relevant tokens in the document rather than all the tokens, thus eliminating the stage of performing ANN.

As another research effort in the stream of multi-vector retrieval approaches, we begin by asking the following questions: 1) Can we make single-stage retrieval possible in a multi-vector retrieval approach? Multi-stage retrieval requires additional ANN training for clustering based on the trained model for queries and documents at the token retrieval stage, the ANN training necessitates optimizing the number of clusters and requires high computing power proportional to the number of tokens in the collection. 2) Can we achieve lightweight indexing while minimizing the loss of contextual in-

*Corresponding Author

Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 760–770
January 19–24, 2025. ©2025 Association for Computational Linguistics

formation? Prior studies (Gao et al., 2021; Li et al., 2023) have managed to implement lightweight indexing by removing document tokens that do not directly match those in the query and by employing an inverted index. Nevertheless, pruning tokens based solely on exact matches or indexed words limits the ability to leverage the full semantic richness of all document tokens. Although Li et al. (2023) compensates for the loss of semantic context through the use of a routing algorithm, it still demands considerable engineering effort and detailed optimization.

We introduce a retrieval framework that utilizes a sequence compressive vector (SCV), processed through a coarse-to-fine vector search in end-to-end strategy. Our key idea involves transforming encoded representations of document tokens into span-level embeddings of arbitrary width, thereby compressing the sequence length. As our model performs indexing based on span representations of documents rather than at the token-level retrievers, the index size and the associated computational latency are significantly reduced. Since the lightweight index can perform million-scale retrieval with GPUs, this framework can load single- and multi-vector indexes simultaneously. Accordingly, our framework performs a coarse-to-fine vector search by initially finding a sufficient number of candidate documents with single-vector retrieval and then directly outputting the top-k relevant documents through multi-vector retrieval, using only a trained model without an external retrieval module at inference time.

Additionally, we enhance our model by employing reranking using a cross-encoder (Urbanek et al., 2019). Our experimental results show that the proposed method outpaces the inverted list approach by a factor of 1.1. The SCV model delivers performance comparable to ColBERT and sets a new standard for the base-sized models with reranking. Our contributions can be summarized in threefold:

- We introduce an efficient multi-vector retriever that utilizes tokens compression to span representations.
- The coarse-to-fine vector search framework can process through an end-to-end strategy in a single stage.
- Our approach is 207 times faster than ColBERT and 4.6 times faster than CITADEL.

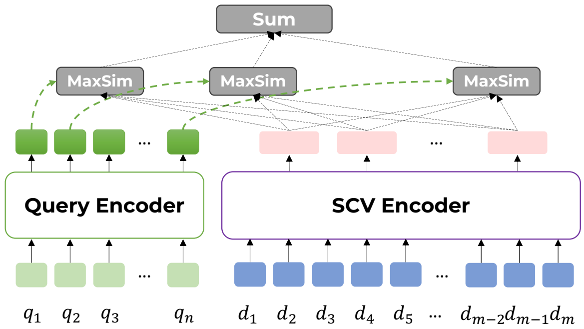

Figure 1: Sequence Compressive Vectors architecture overview.

2 Method

2.1 Preliminaries

The input query is denoted as $Q = \{q_1, q_2, ..., q_n\}$, and the document as $D = \{d_1, d_2, ..., d_m\}$, with the span sequence generated from document tokens represented by $S = \{s_1, s_2, ..., s_l\}$. The n, m, and l are the length of the query, document, and span, respectively. Span sequence is produced using a sliding window algorithm, which maintains context information by allowing overlap of adjacent tokens when extracting tokens within the window. The width of the window is denoted by $W \in \{2, 4, 8, 16\}$, and the interval at which the window moves across tokens, skipping them at a fixed rate, is referred to as $0 \leqslant rate \leqslant 1, rate \in \mathbb{R}$. The overall size of the span sequence is determined by the following equation:

$$l = \left\lceil \frac{m - W}{(1 - rate)W} + 1 \right\rceil \tag{1}$$

2.2 Model Structure

SCV retriever is a multi-vector retrieval model as illustrated in Figure 1. It compresses token information of the document by extracting fixed length spans and allowing the model to train span embeddings. Pre-trained language model (PLM) (Devlin et al., 2019; Sanh et al., 2020), is used to encode the input sequence of the query, $\mathbf{h}_{q_i} = \text{PLM}(q_i)$, and the document, $\mathbf{h}_{d_j} = \text{PLM}(d_j)$, where the language encoders are shared. Special tokens of [Q] and [D] are prefixed to the query and the document, respectively, to differentiate between query and document inputs. Given a document token vector, $\mathbf{h}_{d_j}$, the span level representation is computed as $\mathbf{h}_s = \phi(\mathbf{h}_d)$, where ϕ is a span compressive vector operation. We discuss this operation further in detail in Chapter 2.3.

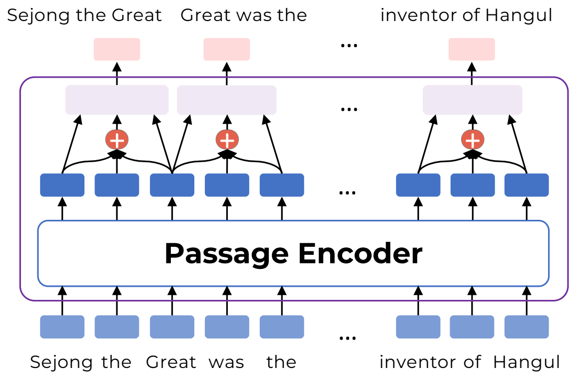

Figure 2: SCV Encoder for Span Representation.

Our model leverages the full contextualized representations of query tokens and document spans. Within the SCV encoder, the compressed document span representations engage with the query token vector via a MaxSim (Khattab and Zaharia, 2020), which is used to calculate the document score. This process is articulated in the equation below:

$$f(Q, S) = \sum_{i=1}^{n} \max_{k=1,\ldots,l} \mathbf{h}_{q_i}^{\intercal} \mathbf{h}_{s_k} \tag{2}$$

where $\mathbf{h}_{q_i}$ and $\mathbf{h}_{s_k}$ denote the last-layer contextualized token embeddings of a query and span embeddings of a document.

Popular retrieval models (Gao et al., 2021; Li et al., 2023) use vectors of a CLS special token in query and document, respectively, to provide high level semantic matching between the query and document. We further leverage the [CLS] vector similarity, representing the aggregate sequence of both the query and document as follows.

$$\begin{aligned} \mathbf{v}_{q_{cls}} &= \mathbf{W}_{cls}\mathbf{h}_q + \mathbf{b}_{cls} \\ \mathbf{v}_{d_{cls}} &= \mathbf{W}_{cls}\mathbf{h}_d + \mathbf{b}_{cls} \end{aligned} \tag{3}$$

2.3 Sequence Compressive Vectors

We introduce an end-to-end retrieval framework designed for multi-vector retrieval, which compresses token sequences from documents as depicted in Figure 2. For example, the process begins with the input sequence being encoded with contextualized token representations through an encoder. With $W = 3$, the model utilizes the sliding window method to extract token representations, subsequently compressing these into span-level information through diverse pooling techniques.

The core idea of SCV lies in the span representation, $\mathbf{h}_s$, with the compression ratio influenced by W and $rate$, as outlined in Equation 1. A feed-forward neural network with an activation function is used to encode lexical information. This encoded information is subsequently concatenated with pooled vectors from document tokens, resulting in the span representation, $\mathbf{h}_s$, for span k:

$$\begin{aligned} \phi(\mathbf{h}_d) &= \mathrm{GELU}(\mathrm{FFNN}(v_{comp})) \\ v_{comp} &= [\mathbf{g}^s; \mathbf{g}^e; \mathbf{g}^m; \mathbf{g}^c; \mathbf{h}^{\mathrm{sum}}_{d_{[j:j+W]}}; \mathbf{h}^{\mathrm{max}}_{d_{[j:j+W]}}; \alpha] \\ \mathbf{g}^s &= \mathrm{FFNN}(\mathbf{h}_{d_j}) \\ \mathbf{g}^e &= \mathrm{FFNN}(\mathbf{h}_{d_{[j+W]}}) \\ \mathbf{g}^m &= \mathbf{g}^s \circ \mathbf{g}^e \\ \mathbf{g}^c &= \mathrm{GELU}(\mathrm{FFNN}([\mathbf{g}^s; \mathbf{g}^e])) \\ \alpha &= \max(\mathrm{attn}(\mathbf{h}_{d_{[j:j+W]}}, \mathbf{h}_{d_{[j:j+W]}})\mathbf{h}_{d_{[j:j+W]}}) \end{aligned} \tag{4}$$

where $\circ$ denotes element-wise multiplication, $\mathbf{h}^{\mathrm{sum}}$ and $\mathbf{h}^{\mathrm{max}}$ are pooled vectors for sum and max pooling, respectively. α represents a salient word using an attention mechanism (Bahdanau et al., 2015), which is highlighted for the most relevant parts of the sequence, and max pooling over words in each span. Max operation involves taking the most important feature (Kim, 2014) and sum operation captures the global intensity of features across the span is relevant (Tian et al., 2017). The above formula generalizes the span representation that includes the start and end boundary representations of the span, as well as the representation of salient words within the span.

2.4 Training

We train SCV using loss of negative log likelihood based on similarity score of $f(Q, S)$ of Equation 2 for a query q, a positive sample d^+, and a set of negative samples $N = \{d_1^-, d_2^-, ..., d_B^-\}$, where B is the batch size. Our strategy involves contrastive learning with a focus on negative sample utilization. We utilize in-batch negatives (ib) (Karpukhin et al., 2020), pre-batch negatives (pb) (Kim et al., 2022), and hard negatives (hb) generated by BM25 (Robertson and Zaragoza, 2009) that are widely used in the retrieval tasks.

$$\mathcal{L} = -\log \frac{\exp(f(q,d^+))}{\exp(f(q,d^+)) + \sum_{b \in N_{\mathrm{ib}} \cup N_{\mathrm{pb}} \cup N_{\mathrm{hb}}} \exp(f(q,d_b^-))} \tag{5}$$

where the numbers of negatives are $|N_{\mathrm{ib}}| = B - 1$, $|N_{\mathrm{pb}}| = B$, and $|N_{\mathrm{hb}}| = H$, H is a hyper-parameter for the number of hard negatives.

We enhance the training of span representation-based retrieval scores between queries and documents by employing multi-task learning with the single vector retriever. Multi-vector retrieval model calculates SCV loss $\mathcal{L}_{SCV}$ and token-level all-to-all retriever loss $\mathcal{L}_{tok}$, respectively, according to Equation 5. Meanwhile, the single vector retrieval computes its loss $\mathcal{L}_{cls}$ by performing a dot-product with the score from Equation 3, and the total loss is obtained by summing all contributions.

The final loss equation is as follows:

$$\mathcal{L} = \mathcal{L}_{SCV} + \mathcal{L}_{tok} + \alpha\mathcal{L}_{cls} \tag{6}$$

where α is used to scale loss of the single vector retriever.

In addition, we augment question synthetic data by prompting MS MARCO passages to GPT-4 [1] to enhance representations of span embeddings. Question generation is sequentially conducted to the passages, producing approximately 180k questions, while ensuring that the development set of MS MARCO remains unseen. We perform lexical filtering and cleaning for the generated questions.

2.5 Coarse-to-fine Vector Search

Even though sequence compression reduces the storage requirements, searching documents still results in increased computation proportional to the index size, leading to latency. To facilitate faster search times, we execute an coarse-to-fine vector search using a single model, as follows: The SCV model calculates dot product using the CLS token vectors for queries and documents and conducts multi-task learning. During inference time, based on the CLS token vectors trained in this manner, we first perform single-vector retrieval to extract the top-N documents, with $N \in \{10000, 20000, 50000, 100000\}$, followed by multi-vector retrieval using the extracted document vectors to produce the top-k final search results. Our framework is end-to-end process and light and fast as it performs model scoring without the need for the external retrieving such as ANN. Following the aforementioned process, we optionally apply re-ranking to enhance the search quality.

[1]https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo

Models	TREC DL 19 nDCG@10	TREC DL 19 R@1k	Index (GB)	Latency (ms/query)
Models trained with only BM25 hard negatives				
BM25	0.506	0.739	0.67	×
DPR-768	0.611	0.742	<u>26</u>	<u>1.28</u>
COIL-tok	0.660	0.809	52.5	46.8
ColBERT	0.694	0.830	154	178
CITADEL	0.687	0.829	78.3	3.95
SCV	0.645	0.712	30	**0.86**
Models trained with further methods				
coCondenser	0.674	0.820	26	1.28
ColBERT-v2	**0.744**	**0.882**	29	122
ColBERT-PLAID	<u>0.744</u>	<u>0.882</u>	**22.1**	55
CITADEL+	0.703	0.830	26.7	3.21

Table 1: In-domain evaluation on TREC DL 2019. Performance reference is made to CITADEL, and latency includes the total time for query encoding and search.

3 Experimental Results

We train our model using the passage ranking dataset from MS MARCO [2]. For in-domain evaluation, we use the MS MARCO development set and TREC DL 2019, and for out-of-domain evaluation, we assess performance on the BEIR benchmark (Thakur et al., 2021). The MS MARCO development set contains 6,980 queries, while the TREC DL 2019 evaluation set provides annotations for 43 queries. The BEIR benchmark comprises 18 retrieval tasks across 9 domains, and we evaluate using 13 datasets following previous studies (Santhanam et al., 2022a; Li et al., 2023).

As our evaluation metric, we employ nDCG@10, and Recall@1000 for MS MARCO, along with nDCG@10 for BEIR. We use a script of BEIR [3] to evaluate datasets.

Experimental settings We initialize using DistilBERT-base (Sanh et al., 2019) as our backbone model. The experimental environment for training, indexing, and retrieval utilizes a Tesla A100 GPU, with an optimized batch size set to 630. Evaluation during training is conducted with in-batch predictions of size 1k, and checkpoints are saved at the step showing the best performance. The SCV model is trained using the AdamW (Loshchilov and Hutter, 2017) optimizer, with a learning rate of $5e-5$ and linear scheduling. Hard negatives are sampled from the top 1000 BM25 results (Gao et al., 2023), and each query

[2]https://github.com/microsoft/MS MARCO-Passage-Ranking

[3]https://github.com/beir-cellar/beir

Methods	AA	CF	DB	Fe	FQ	HQ	NF	NQ	Qu	SF	SD	TC	T2	Avg.
BM25	0.315	0.213	0.313	0.753	0.236	0.603	**0.325**	0.329	0.789	0.665	0.158	0.656	**0.367**	0.440
DPR-768	0.323	0.167	0.295	0.651	0.224	0.441	0.244	0.410	0.750	0.479	0.103	0.604	0.185	0.375
ColBERT	0.233	0.184	0.392	0.771	0.317	0.593	0.305	**0.524**	0.854	0.671	**0.165**	0.677	0.202	0.453
GTR	**0.511**	**0.215**	0.392	0.660	**0.349**	0.535	0.308	0.495	**0.881**	0.600	0.149	0.539	0.215	0.452
CITADEL	0.503	0.191	**0.406**	**0.784**	0.298	**0.653**	0.324	0.510	0.844	**0.674**	0.152	**0.687**	0.294	**0.486**
SCV	0.464	0.139	0.351	0.675	0.272	0.535	0.315	0.425	0.774	0.656	0.135	0.668	0.262	0.436

Table 2: nDCG@10 on BEIR. Dataset Legend (Li et al., 2023): AA=ArguAna, CF=Climate-FEVER, DB=DBPedia, Fe=FEVER, FQ=FiQA, HQ=HotpotQA, NF=NFCorpus, NQ=NaturalQuestions, Qu=Quora, SF=SciFact, SD=SCIDOCS, TC=TREC-COVID, T2=Touché.

uses 1 positive and 1 negative sample. The dimension size for both the CLS token layer and the SCV output layer is set to 128. During training, the width of span embeddings (W) is set to 8, while for indexing, it is adjusted to 16 for MS MARCO and remains at 8 for BEIR. The sliding overlap rate ($rate$) is 0.2, the dimension size for span embeddings is 384, and the dropout rate is set to 0.1. In Chapter 2.5, it is mentioned that inference is performed with N set to 10k. All hyper-parameters are optimized.

3.1 Results

Results on MS MARCO Table 1 presents the performance on in-domain datasets along with index storage size and search latency. The comparison models utilize BM25 hard negatives or include further pre-training, hard-negative mining, and distillation for training, such as coCondenser (Gao and Callan, 2022), ColBERT-v2 (Santhanam et al., 2022c), and ColBERT-PLAID (Santhanam et al., 2022b). The experimental results show that while our SCV method achieves comparable performance to other models on TREC DL 19 using only BM25 hard negatives. In contrast, SCV's index size is a more compact 30GB, close to DPR-768, and reduces the size by approximately 5.13 times compared to ColBERT.

SCV achieves a latency of 0.86 ms/query, making it the fastest among the multi-vector retrieval models, and approximately 3.7 times, 64 times, 141.8 times, and 207 times faster than CITADEL+, ColBERT-PLAID, ColBERT-v2, and ColBERT, respectively. Furthermore, our framework is approximately 1.5 times faster than the single vector retriever DPR-768. Most RAG or question answering pipeline services use single vector retriever due to processing speed issues. We expect that our approach can provide a faster and more accurate retrieval model for these systems.

Models	Size	TREC DL 19 nDCG@10
Reranking models		
monoBERT (Nogueira et al., 2019)	110M	0.723
SimLM (Wang et al., 2023)	110M	0.741
ListT5 (Yoon et al., 2024)	220M	0.718
SCV+CE	220M	**0.744**
Ranking models with LLM		
RankLLaMA (Ma et al., 2024)	7B	0.756
RankLLaMA	13B	**0.760**
RankVicuna (Pradeep et al., 2023)	7B	0.668
PRP (Qin et al., 2024)	20B	0.727

Table 3: In-domain Reranking evaluation on TREC DL 2019. Performance reference is made to RankLLaMA.

Results on BEIR We conduct an out-of-domain evaluation using the BEIR benchmark. Table 2 presents the zero-shot evaluation results on BEIR for retrieval models, including those extended with re-ranking. The experimental outcomes demonstrate that the SCV significantly outperforms a single-vector retriever and is competitive with multi-vector retrievers. SCV utilizes a compressed representation of span to generate multi-vector from token sequences, we expect its performance to fall between that of DPR and ColBERT. According to the experimental results, SCV shows scores close to the ColBERT, as we expected and specifically achieves higher scores on the AA, NF, and T2 datasets.

Results with Reranker To further enhance performance, we conducted reranking using the cross-encoder (CE) version ms-marco-MiniLM-L-6-v2 based on the SCV retrieval results. In contrast, all comparison models in Table 3 performed reranking based on BM25 retrieval results. The SCV+CE pipeline achieved an nDCG of 0.744 on TREC DL 19, showing an improvement of 0.099 in nDCG compared to the SCV retriever in Table 3. This result is 0.21 higher than monoBERT, indicating that retrieving relevant candidates during the retrieval stage positively impacts reranking. Moreover, it is evident that reranking using the proposed method outperforms relatively recent studies such as SimLM and ListT5.

Unlike the previous experimental setup, the results in the following row are based on reranking using LLMs. The LLM approach involves decoder-only variations, with model sizes including 7B, 13B, and 20B. In reranking, RankLLaMA-13B demonstrated the best performance, followed by RankLLaMA-7B and the PRP model. Overall, LLM-based models exhibited higher performance compared to methods using small language models (SLM) as the backbone, but the differences in model size were quite significant. Despite PRP having the largest scale with a model size of 20B among the LLM-based methods, it showed relatively lower performance and lacked competitiveness against SLM backbone models. Therefore, in in-domain retrieval, a well-tuned combination of small retrieval and ranking models remains competitive compared to LLM-based ranking models.

4 Related Works

Modern RAG with Retriever Recently, with the advent of LLMs, there has been significant development and study related to RAG pipelines. Study on the RAG framework includes not only methods to enhance LLM performance but also attempts to refine performance based on retrieval results. This includes methods for summarizing retrieved results (Kim et al., 2024) and creating new retrieval results (Asai et al., 2024). Shao et al. (2023) generates responses by re-retrieving chunks based on the retrieved chunks and generated results. Shi et al. (2024) enhances the retriever to improve the performance of the LM based on the RAG structure.

Neural Information Retrieval Deep language models have significantly influenced neural information retrieval. A prevalent method involves processing the query-document pair with BERT, using the output of BERT's [CLS] token to determine a relevance score (Karpukhin et al., 2020). (Khattab and Zaharia, 2020) represents document text as a collection of token rather than a single vector and apply late interaction between the document and the query, implementing a late interaction mechanism between the document and the query. This method enables comprehensive semantic and lexical matching between queries and documents, reaching state-of-the-art performance across numerous benchmarks. Yet, the scalability of their non-linear scoring function faces challenges when extended to millions of documents. Alternative strategies (Gao et al., 2021; Li et al., 2023; Lee et al., 2023) simplify the multi-vector retrieval by focusing on retrieving only the most relevant tokens for ranking candidates, effectively pruning the document tokens.

Span Representation Span representation has primarily been utilized in information extraction tasks for processing documents. (Lee et al., 2017) enables end-to-end coreference resolution by extracting span representations and ranking span pairs. Performance improves significantly when BERT is adapted to whole word masking, leading to the development of SpanBERT (Joshi et al., 2020), which trains the model by setting the mask token unit to spans. SpanBERT helps to span-based approaches. In nested named entity recognition tasks (Zhu et al., 2023; Zhu and Li, 2022; Wan et al., 2022; Zhang et al., 2023), span representation is employed to address the problem by handling the range of chunks that are entities through span-based modeling and attaching entity tags.

5 Conclusion

In this paper, we propose an end-to-end multi-vector retrieval framework utilizing sequence compression, named SCV. Our method achieves a latency of 0.8 ms/query when querying a million-scale index, which is 207 times faster than ColBERT and 4.6 times faster than the fastest multi-vector retriever, CITADEL, on GPUs. While SCV records performance comparable to other multi-vector retrieval models, its major strength lies in its very small latency. Leveraging this advantage for re-ranking, SCV achieves state-of-the-art results among other SLM-based ranking models and shows promise among re-ranking methods. Our model minimizes information loss in the document sequence by fully utilizing token information to create span representations. Compressing token vectors has a strong potential of more efficiently and effectively model retrieval tasks.

Finally, in the modern RAG, additional modules are configured, including not only retrieval and generation but also the use of retrieval, retrieval summarization, and iterative retrieval. We believe that as more of these components are added, the speed of retrieval becomes increasingly important in real-world services.

6 Limitations

The proposed RAG system is designed to be more suitable for practical service use, focusing on the speed of the RAG system. As a result, there may be a slight performance decline compared to existing SOTA models. However, implementing this algorithm into an operational system is not technically difficult, so there is potential to maximize its usability based on the code that will be released in the future.

References

Raviteja Anantha and Danil Vodianik. 2024. Context tuning for retrieval augmented generation. In Proceedings of the 1st Workshop on Uncertainty-Aware NLP (UncertaiNLP 2024), pages 15–22, St Julians, Malta. Association for Computational Linguistics.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In The Twelfth International Conference on Learning Representations.

Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023. Direct fact retrieval from knowledge graphs without entity linking. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10038–10055, Toronto, Canada. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, page 2288–2292, New York, NY, USA. Association for Computing Machinery.

Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2843–2853, Dublin, Ireland. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3030–3042, Online. Association for Computational Linguistics.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Tevatron: An efficient and flexible toolkit for neural retrieval. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23, page 3120–3124, New York, NY, USA. Association for Computing Machinery.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. Transactions of the Association for Computational Linguistics, 8:64–77.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Gyuwan Kim, Jinhyuk Lee, Barlas Oguz, Wenhan Xiong, Yizhe Zhang, Yashar Mehdad, and William Yang Wang. 2022. Bridging the training-inference gap for dense phrase retrieval. In Findings of the Association for Computational Linguistics: EMNLP 2022, pages 3713–3724, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain QA of LLMs. In The Twelfth International Conference on Learning Representations.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Y. Zhao. 2023. Rethinking the role of token retrieval in multi-vector retrieval. CoRR, abs/2304.01982.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Minghan Li, Sheng-Chieh Lin, Barlas Oguz, Asish Ghoshal, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. CITADEL: Conditional token interaction via dynamic lexical routing for efficient and effective multi-vector retrieval. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11891–11907, Toronto, Canada. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24, page 2421–2425, New York, NY, USA. Association for Computing Machinery.

Craig Macdonald and Nicola Tonellotto. 2021. On approximate nearest neighbour selection for multi-stage dense retrieval. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, page 3318–3322, New York, NY, USA. Association for Computing Machinery.

C.D. Manning, P. Raghavan, and H. Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rodrigo Frassetto Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. CoRR, abs/1910.14424.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. Preprint, arXiv:2309.15088.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. In Findings of the Association for Computational Linguistics: NAACL 2024, pages 1504–1518, Mexico City, Mexico. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. Found. Trends Inf. Retr., 3(4):333–389.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. Preprint, arXiv:1910.01108.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. Plaid: An efficient engine for late interaction retrieval. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM '22, page 1747–1756, New York, NY, USA. Association for Computing Machinery.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022b. Plaid: An efficient engine for late interaction retrieval. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM '22, page 1747–1756, New York, NY, USA. Association for Computing Machinery.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022c. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 9248–9274, Singapore. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. RE-PLUG: Retrieval-augmented black-box language models. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.

Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. 2017. How to make context more useful? an empirical study on context-aware neural conversational models. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 231–236, Vancouver, Canada. Association for Computational Linguistics.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 673–683, Hong Kong, China. Association for Computational Linguistics.

Juncheng Wan, Dongyu Ru, Weinan Zhang, and Yong Yu. 2022. Nested named entity recognition with span-level graphs. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 892–903, Dublin, Ireland. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023. SimLM: Pre-training with representation bottleneck for dense passage retrieval. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2244–2258, Toronto, Canada. Association for Computational Linguistics.

Soyoung Yoon, Eunbi Choi, Jiyeon Kim, Hyeongu Yun, Yireun Kim, and Seung won Hwang. 2024. Listt5: Listwise reranking with fusion-in-decoder improves zero-shot retrieval. Preprint, arXiv:2402.15838.

Sheng Zhang, Hao Cheng, Jianfeng Gao, and Hoifung Poon. 2023. Optimizing bi-encoder for named entity recognition via contrastive learning. Preprint, arXiv:2208.14565.

Enwei Zhu and Jinpeng Li. 2022. Boundary smoothing for named entity recognition. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7096–7108, Dublin, Ireland. Association for Computational Linguistics.

Enwei Zhu, Yiyang Liu, and Jinpeng Li. 2023. Deep span representations for named entity recognition. In Findings of the Association for Computational Linguistics: ACL 2023, pages 10565–10582, Toronto, Canada. Association for Computational Linguistics.

A Appendix

A.1 Applied Hyperparameters

	value
Backborn	DistilBERT-base
Optimizer	AdamW
learning_rate	5.0e-5
Dropout	0.05
lr_scheduler	cosine
Epoch	10
sequence_len	512
Batch size	630
Random Seed	1004
BM25 TOP n	1000

Table 4: Applied hyperparameter settings.

A.2 Details of Experimental Environments

The hyperparameter settings used in this study can be found in 4. The model essentially adopts the DistilBERT-base model, and experiments were conducted based on the top 1000 search results retrieved by BM25. The batch size was set to 630, utilizing the maximum size available on an A100 GPU. Specific learning rates and token sizes are provided in Table 4.

A.3 Coarse-to-fine Search Overview

SCV employs multi-task learning to jointly train single-vector and multi-vector retrieval. During the indexing phase, the [CLS] token vector is stored with span-level vectors for each document in the collection. At inference time, the SCV model retrieves the stored single vector and span vectors for each document, loading them into memory. An overview of this process is presented in Fig. 4, specifically in the `On memory` section. In the `Process` section, the [CLS] token vector of each document, loaded into memory, is used to compute similarity with the [CLS] vector of the encoded query. The top N relevant document IDs are then selected. Without additional gathering operations, the system directly computes the maximum similarity between query token vectors and document span vectors, ultimately producing the top K relevant document IDs. This approach eliminates the need for intermediate gathering operations, enabling a coarse-to-fine retrieval process. It efficiently identifies candidate relevant documents at a coarse level and performs fine-grained token- and span-level retrieval based on these candidates in an end-to-end manner. Compared to traditional two-stage methods, SCV offers a simpler and faster way to retrieve relevant documents.

A.4 Prompt template

We use GPT-4 for question augmentation. The prompt used for augmentation is shown in Figure 3, and passages from MSMARCO are randomly sampled and input along with the prompt.

# Num. of Q	nDCG@100	Recall@100
w/o aug.	0.305	0.267
50k	0.301	0.265
100k	0.275	0.253
150k	**0.315**	**0.278**
200k	0.300	0.266

Table 5: Ablation for question augmentation

A.5 Ablation for Query Augmentation

To make the model more robust by learning diverse expressions for the retriever's positive samples, we perform question augmentation using GPT-4. Table 5 shows the performance changes with the use of augmented questions. We create augmentation amounts of 50k, 100k, 150k, and 200k, and among these, using 150k results in the best performance.

A.6 Reranking Result for Out-of-domain

In Table 6, we measure the reranking performance on out-of-domain data using the BIER benchmark.

Leveraging the advantage of SCV's rapid latency, we perform a re-ranking on the top-1000 retrieval results. Compared to BM25+CE using the same re-ranking model, our approach exhibits superior performance, indicating its efficacy in identifying candidate documents for zero-shot scenarios.

The experimental results show that the performance of the SCV retrieval stage is 0.436 according to Table 2, and reranking improves the score by 0.073. Although it shows a lower average score compared to HYRR or RankT5-large, it is improved compared to BM25+CE, which uses the same ranking model CE.

Please provide a high-quality answer to the part I requested. Take a deep breath and think slowly. Create as many questions as possible, over 20, using only the content included in the input document. Base the questions on 'when, where, what, why, who (or what), and how'. Gradually think and create questions of various types such as 'comparison, fact verification, quantity, keyword, conversational, domain-specific', etc. For question generation, use [G] as a delimiter to insert one question at a time, and indicate whether the answer to the generated question can be found in the input paragraph with [sufficient|average|insufficient|none]. To summarize the request, everything is in Korean, and the task is to create questions dependent on the given document. You are a child with a lot of knowledge. You can think of a wide variety of questions for a single entity. So, create various questions that can be made from the above document for me. Focus on questions that people would ask via web search or phone calls. Avoid vague questions that ask about articles or pronouns like 'this' or 'that'. And only create questions whose answers can be found in the given document. I will enter the document as [D].
[D]: {Input passage}

Figure 3: Prompt template design for question generation.

Methods	AA	CF	DB	Fe	FQ	HQ	NF	NQ	Qu	SF	SD	TC	T2	Avg.
BM25+CE	0.311	0.253	0.409	0.819	0.347	0.707	0.350	0.533	0.825	0.688	0.166	0.757	0.271	0.495
HYRR	0.344	**0.272**	0.385	**0.868**	0.408	0.706	0.379	0.532	**0.861**	0.734	**0.183**	0.796	0.368	**0.526**
RankT5-large	0.330	0.215	0.442	0.832	**0.445**	**0.710**	**0.381**	**0.614**	0.831	**0.750**	0.181	**0.807**	**0.440**	0.524
SCV+CE	**0.508**	0.240	**0.452**	0.804	0.365	0.691	0.339	0.570	0.826	0.673	0.164	0.720	0.267	0.509

Table 6: nDCG@10 on BEIR. Dataset Legend is same to Table 2.

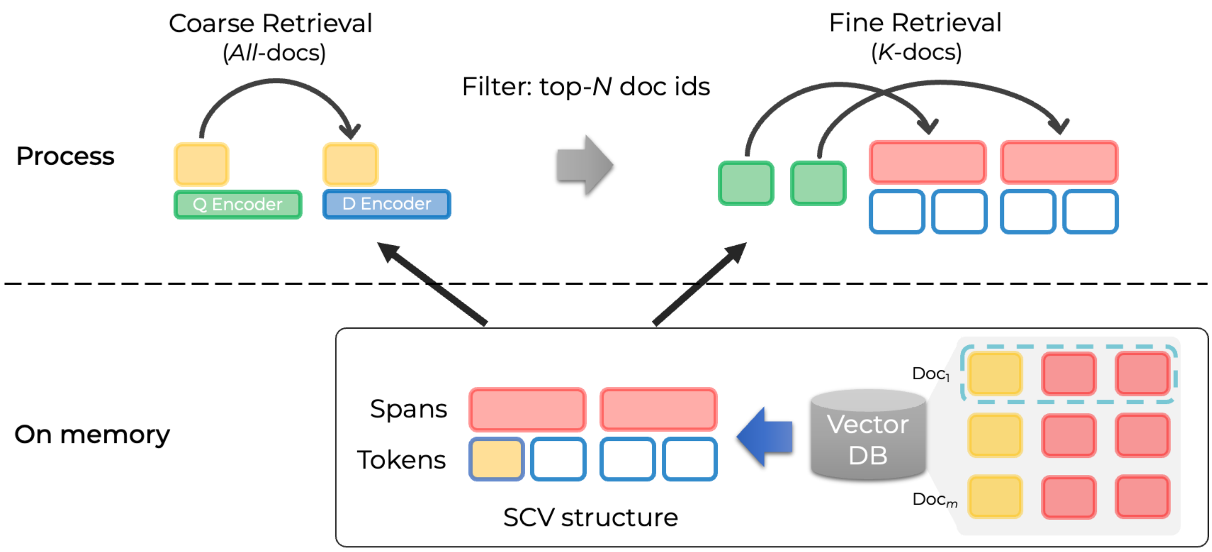

Figure 4: Coarse-to-fine search overview. In the figure, yellow boxes represent the vectors of a single-vector retriever, while red boxes denote the vectors of individual spans. The empty boxes outlined in blue indicate token-level vectors for SCV but are not used during model runtime. The green box illustrates the abstract structure of the Q Encoder for questions, and the blue box represents the D Encoder for documents.