CITADEL: Conditional Token Interaction via Dynamic Lexical Routing for Efficient and Effective Multi-Vector Retrieval

Minghan Li[1][*]**, Sheng-Chieh Lin**[1]**, Barlas Oguz**[2]**, Asish Ghoshal**[2]**,
Jimmy Lin**[1]**, Yashar Mehdad**[2]**, Wen-tau Yih**[2]**, and Xilun Chen**[2][†]

University of Waterloo[1], Meta AI[2]

{m692li,s269lin,jimmylin}@uwaterloo.ca

{barlaso,aghoshal,mehdad,scottyih,xilun}@meta.com

Abstract

Multi-vector retrieval methods combine the merits of sparse (e.g. BM25) and dense (e.g. DPR) retrievers and achieve state-of-the-art performance on various retrieval tasks. These methods, however, are orders of magnitude slower and need more space to store their indexes compared to their single-vector counterparts. In this paper, we unify different multi-vector retrieval models from a token routing viewpoint and propose conditional token interaction via dynamic lexical routing, namely CITADEL, for efficient and effective multi-vector retrieval. CITADEL learns to route each token vector to the predicted lexical "keys" such that a query token vector only interacts with document token vectors routed to the same key. This design significantly reduces the computation cost while maintaining high accuracy. Notably, CITADEL achieves the same or slightly better performance than the previous state of the art, ColBERT-v2, on both in-domain (MS MARCO) and out-of-domain (BEIR) evaluations, while being nearly **40 times** faster. Source code and data are available at https://github.com/facebookresearch/dpr-scale/tree/citadel.

1 Introduction

The goal of information retrieval (Manning et al., 2008) is to find a set of related documents from a large data collection given a query. Traditional bag-of-words systems (Robertson and Zaragoza, 2009; Lin et al., 2021a) calculate the ranking scores based on the query terms appearing in each document, which have been widely adopted in many applications such as web search (Nguyen et al., 2016; Noy et al., 2019) and open-domain question answering (Chen et al., 2017; Lee et al., 2019). Recently, dense retrieval (Karpukhin et al., 2020) based on

---

[*]This work is done during Minghan's internship at Meta.
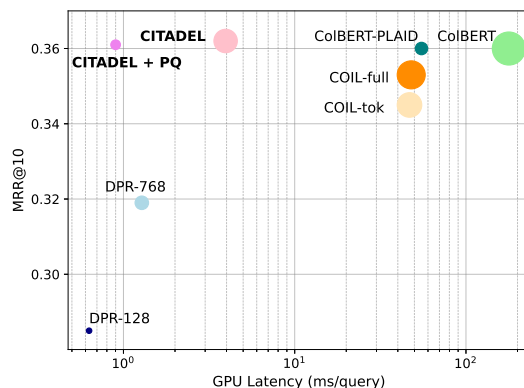[†]Xilun and Minghan contributed equally to this work.

Figure 1: GPU latency vs ranking quality (MRR@10) on MS MARCO passages with an A100 GPU. The *circle size* represents the relative index storage on disk. All models are trained without hard-negative mining, distillation, or further pre-training.

pre-trained language models (Devlin et al., 2019; Liu et al., 2019) has been shown to be very effective. It circumvents the term mismatch problem in bag-of-words systems by encoding the queries and documents into low-dimensional embeddings and using their dot product as the similarity score (Figure 2a). However, dense retrieval is less robust on entity-heavy questions (Sciavolino et al., 2021) and out-of-domain datasets (Thakur et al., 2021), therefore calling for better solutions (Formal et al., 2021b; Gao and Callan, 2022).

In contrast, multi-vector retrieval has shown strong performance on both in-domain and out-of-domain evaluations by taking into account token-level interaction. Among them, ColBERT (Khattab and Zaharia, 2020) is arguably the most celebrated method that has been the state of the art on multiple datasets so far. However, its wider application is hindered by its large index size and high retrieval latency. This problem results from the redundancy in the token interaction of ColBERT where many tokens might not contribute to the sentence semantics at all. To improve this, COIL (Gao et al., 2021a) imposes an exact match constraint on ColBERT
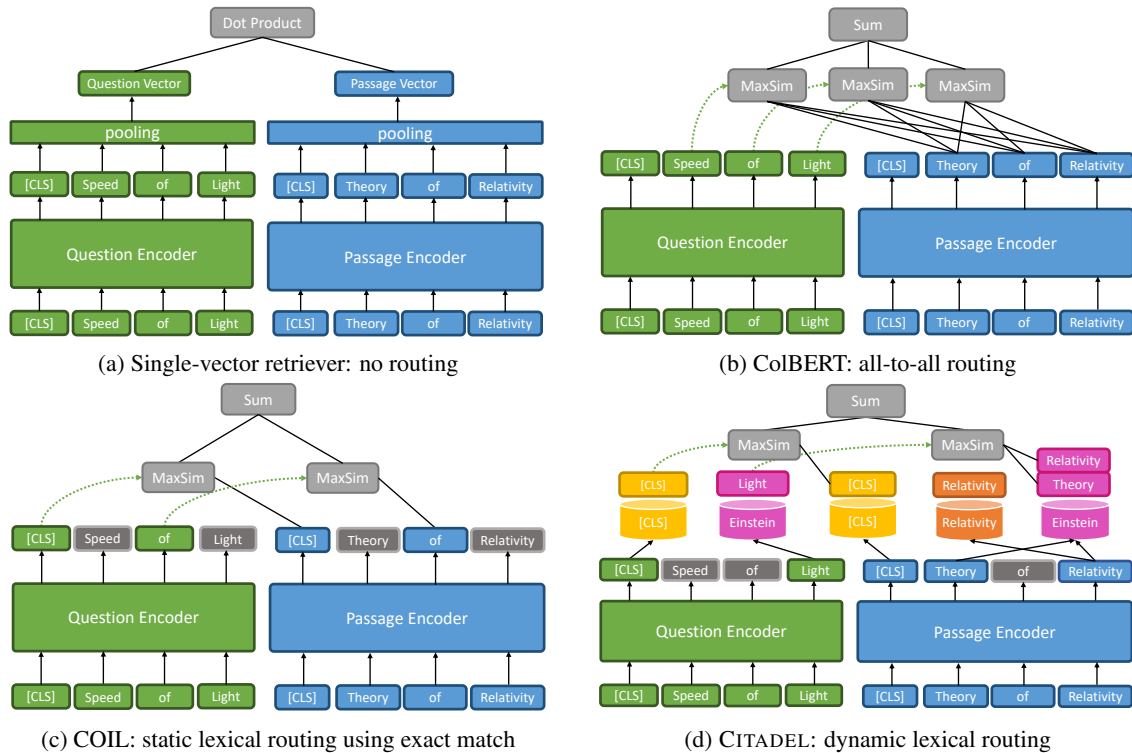
11891

Figure 2: A unified token routing view over different multi-vector and single-vector retrieval models. The cylinder in (d) represents the lexical key for token routing. The grey tokens in (c) and (d) represent the tokens that do not contribute to the final similarity. For (d), CITADEL routes "*light*", "*theory*", and "*relativity*" to the "*Einstein*" key to avoid term mismatch using a learned routing function.

for conditional token interaction, where only token embeddings with the same token id could interact with each other. Although reducing the latency, the word mismatch problem reoccurs and the model may fail to match queries and passages that use different words to express the same meaning.

In this paper, we first give a unified view of existing multi-vector retrieval methods based on *token routing* (Section 2), providing a new lens through which we expose the limitations of current models. Under the token routing view, ColBERT could be seen as *all-to-all routing*, where each query token exhaustively interacts with all passage tokens (Figure 2b). COIL, on the other hand, could be seen as *static lexical routing* using an exact match constraint, as each query token only interacts with the passage tokens that have the same token id as the query token (Figure 2c).

In contrast, we propose a novel conditional token interaction method using *dynamic lexical routing* called CITADEL as shown in Figure 2d. Instead of relying on static heuristics such as exact match, we train our model to dynamically moderate token interaction so that each query token only interacts with the most relevant tokens in the passage. This is achieved by using a lexical *router*, trained end-

to-end with the rest of the model, to route each contextualized token embedding to a set of activated lexical "keys" in the vocabulary. In this way, each query token embedding only interacts with the passage token embeddings that have the same activated key, which is dynamically determined during computation. As we shall see in Section 5.1, this learning-based routing does not lose any accuracy compared to all-to-all routing while using fewer token interactions than COIL (Section 3.4), leading to a highly effective and efficient retriever.

Experiments on MS MARCO passages (Nguyen et al., 2016) and TREC DL show that CITADEL achieves the same level of accuracy as ColBERT-v2. We further test CITADEL on BEIR (Thakur et al., 2021) and CITADEL still manages to keep up with ColBERT-v2 (Santhanam et al., 2022b) which is the current state of the art. As for the latency, CITADEL can yield an average latency of 3.21 ms/query on MS MARCO passages using an A100 GPU, which is nearly 40× faster than ColBERT-v2. By further combining with product quantization, CITADEL's index only takes 13.3 GB on MS MARCO passages and reduces the latency to 0.9 ms/query as shown in Figure 1.

2 A Unified Token Routing View of Multi-Vector Retrievers

We outline a unified view for understanding various neural retrievers using the concept of token routing that dictates token interaction.

2.1 Single-Vector Retrieval

Given a collection of documents and a set of queries, single-vector models (Karpukhin et al., 2020; Izacard et al., 2022) use a bi-encoder structure where its query encoder $\eta_Q(\cdot)$ and document encoder $\eta_D(\cdot)$ are independent functions that map the input to a low-dimensional vector. Specifically, the similarity score s between the query q and document d is defined by the dot product between their encoded vectors $v_q = \eta_Q(q)$ and $v_d = \eta_D(d)$:

$$s(q, d) = v_q^T v_d. \qquad (1)$$

As all the token embeddings are pooled before calculating the similarity score, no routing is committed as shown in Figure 2a.

2.2 Multi-Vector Retrieval

ColBERT (Khattab and Zaharia, 2020) proposes late interaction between the tokens in a query $q = \{q_1, q_2, \cdots, q_N\}$ and a document $d = \{d_1, d_2, \cdots, d_M\}$:

$$s(q, d) = \sum_{i=1}^{N} \max_{j} v_{q_i}^T v_{d_j}, \qquad (2)$$

where v_{q_i} and v_{d_j} denotes the last-layer contextualized token embeddings of BERT. This is known as the MaxSim operation which exhaustively compares each query token to all document tokens. We refer to this as *all-to-all routing* as shown in Figure 2b. The latency of ColBERT is inflated by the redundancy in the all-to-all routing, as many tokens do not contribute to the sentence semantics. This also drastically increases the storage, requiring complex engineering schemes to make it more practical (Santhanam et al., 2022b,a).

Another representative multi-vector approach known as COIL (Gao et al., 2021a) proposes an exact match constraint on the MaxSim operation where only the embeddings with the same token id could interact with each other. Let $\mathcal{J}_i = \{j \mid d_j = q_i, 1 \leq j \leq M\}$ be the subset of document tokens $\{d_j\}_{j=1}^M$ that have the same token ID as query token

q_i, then we have:

$$s(q, d) = \sum_{i=1}^{N} \max_{j \in \mathcal{J}_i} v_{q_i}^T v_{d_j}, \qquad (3)$$

It could be further combined with Equation (1) to improve the effectiveness if there's no word overlap between the query and documents.

$$s(q, d) = v_q^T v_d + \sum_{i=1}^{N} \max_{j \in \mathcal{J}_i} v_{q_i}^T v_{d_j}. \qquad (4)$$

We refer to this token interaction as *static lexical routing* as shown in Figure 2c. As mentioned in Section 1, the word mismatch problem could happen if $\mathcal{J}_i = \varnothing$ for all q_i, which affects the retrieval accuracy. Moreover, common tokens such as "*the*" will be frequently routed, which will create much larger token indexes compared to those rare words. This bottlenecks the search latency as COIL needs to frequently iterate over large token indexes.

3 The CITADEL Method

3.1 Dynamic Lexical Routing

Instead of using the wasteful all-to-all routing or the inflexible heuristics-based static routing, we would like our model to dynamically select which query and passage tokens should interact with each other based on their contextualized representation, which we refer to as *dynamic lexical routing*. Formally, the routing function (or *router*) routes each token to a set of lexical **keys** in the vocabulary and is defined as $\phi : \mathbb{R}^c \to \mathbb{R}^{|\mathcal{V}|}$ where c is the embedding dimension and $\mathcal{V}$ is the lexicon of keys. For each contextualized token embedding, the router predicts a scalar score for each key in the lexicon indicating how relevant each token is to that key. Given a query token embedding v_{q_i} and a document token vector v_{d_j}, the token level router representations are $w_{q_i} = \phi(v_{q_i})$ and $w_{d_j} = \phi(v_{d_j})$, respectively. The elements in the router representations are then sorted in descending order and truncated by selecting the top-K query keys and top-L document keys, which are $\{e_{q_i}^{(1)}, e_{q_i}^{(2)}, \cdots, e_{q_i}^{(K)}\}$ and $\{e_{d_j}^{(1)}, e_{d_j}^{(2)}, \cdots, e_{d_j}^{(L)}\}$ for q_i and d_j, respectively. In practice, we use K=1 and L=5 as the default option which will be discussed in Section 3.5 and Section 7. The corresponding routing weights for q_i and d_j are $\{w_{q_i}^{(1)}, w_{q_i}^{(2)}, \cdots, w_{q_i}^{(K)}\}$ and $\{w_{d_j}^{(1)}, w_{d_j}^{(2)}, \cdots, w_{d_j}^{(L)}\}$, respectively. The final similarity score is similar to Equation (3), but

we substitute the static lexical routing subset $\mathcal{J}_i$ with a dynamic key set predicted by the router: $\mathcal{E}_i^{(k)} = \{j, l \mid e_{d_j}^{(l)} = e_{q_i}^{(k)}, 1 \leq j \leq M, 1 \leq l \leq L\}$ for each key $e_{q_i}^{(k)}$ of the query token q_i:

$$s(q, d) = \sum_{i=1}^{N} \sum_{k=1}^{K} \max_{j, l \in \mathcal{E}_i^{(k)}} (w_{q_i}^{(k)} \cdot v_{q_i})^T (w_{d_j}^{(l)} \cdot v_{d_j}), \quad (5)$$

Optionally, all [CLS] tokens can be routed to an additional *semantic* key to complement our learned lexical routing. We then follow DPR (Karpukhin et al., 2020) to train the model contrastively. Given a query q, a positive document d^+, and a set of negative documents D^-, the constrastive loss is:

$$\mathcal{L}_e = -\log \frac{\exp(s(q, d^+))}{\exp(s(q, d^+)) + \sum_{d^- \in D^-} \exp(s(q, d^-))}, \quad (6)$$

such that the distance from the query to the positive document d^+ is smaller than the query to the negative document d^-.

3.2 Router Optimization

To train the router representation $\phi(q)$ and $\phi(d)$, we adopt a contrastive loss such that the number of overlapped keys between a query and documents are large for positive (q, d^+) pairs and small for negative pairs (q, d^-). We first pool the router representation for each query and document over the tokens. Given a sequence of token-level router representations $\{\phi(v_1), \phi(v_2), \cdots, \phi(v_M)\}$, the sequence-level representation is defined as:

$$\Phi = \max_{j=1}^{M} \phi(v_j), \quad (7)$$

where the max operator is applied element-wise. Similar to (Formal et al., 2021a), We find max pooling works the best in practice compared to other pooling methods. Subsequently, the contrastive loss for training the router is:

$$\mathcal{L}_r = -\log \frac{\exp(\Phi_q^T \Phi_{d+})}{\exp(\Phi_q^T \Phi_{d+}) + \sum_{d^- \in D^-} \exp(\Phi_q^T \Phi_{d-})}. \quad (8)$$

In addition, we follow SPLADE (Formal et al., 2021b,a) to initialize the router with the pre-trained Masked Language Modelling Layer (MLM). Without proper initialization, it is difficult to optimize the router due the large lexical space and sparse activation. With the pre-trained MLM initialization, the router expands words with similar semantic meaning to sets of keys with large overlap at the beginning of training, making the contrastive loss easier to optimize.

3.3 Sparsely Activated Router Design

Softmax activation is commonly used for computing the routing weights in conditional computation models (Fedus et al., 2022; Mustafa et al., 2022). However, softmax often yields a small probability over a large number of dimensions (in our case, about 30,000) and the product of two probability values are even smaller, which makes it not suitable for yielding the routing weights $w_{q_i}^{(k)}$ and $w_{d_j}^{(l)}$ in Equation (5) as the corresponding gradients are too small. Instead, we use the activation from SPLADE to compute the router representation for a token embedding v_j:

$$\phi(v_j) = \log(1 + \text{ReLU}(W^T v_j + b)), \quad (9)$$

where W and b are the weights and biases of the Masked Language Modeling (MLM) layer of BERT. The SPLADE activation brings extra advantages as the ReLU activation filters irrelevant keys while the log-saturation suppresses overly large "wacky" weights (Mackenzie et al., 2021).

3.4 Regularization for Routing

ℓ_1 **Regularization.** Routing each token to more than one key increases the overall size of the index. Therefore, we propose to use ℓ_1 regularization on the router representation to encourage the router to only keep the most meaningful token interaction by pushing more routing weights to 0:

$$\mathcal{L}_s = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{T} \sum_{k=1}^{|\mathcal{V}|} \phi(v_{ij})^{(k)}, \quad (10)$$

where $|\mathcal{V}|$ is the number of keys, B is the batch size, and T is the sequence length. As shown in Figure 6, CITADEL has a sparsely activated set of keys, by routing important words to multiple lexical keys while ignoring many less salient words, leading to effective yet efficient retrieval.

Load Balancing. As mentioned in Section 2.2, the retrieval latency of COIL is bottlenecked by frequently searching overly large token indexes. This results from the static lexical routing where common "keys" have a larger chance to be activated, which results in large token indexes during indexing. Therefore, a vital point for reducing the latency of multi-vector models is to evenly distribute each token embedding to different keys. Inspired by Switch Transformers (Fedus et al., 2022), we

propose to minimize the load balancing loss that approximates the expected "evenness" of the number of tokens being routed to each key:

$$\mathcal{L}_{\mathrm{b}} = \sum_{k=1}^{|\mathcal{V}|} f_k \cdot p_k, \qquad (11)$$

p_k is the batch approximation of the marginal probability of a token vector routed to the k-th key:

$$p_k = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{T} \frac{\exp(W_k^T v_{ij} + b_k)}{\sum_{k'} \exp(W_{k'}^T v_{ij} + b_{k'})}, \qquad (12)$$

where W and b are the weights and bias of the routing function in Equation (9) and v_{ij} is the j-th token embedding in sample i of the batch. f_k is the batch approximation of the total number of tokens being dispatched to the k-th key:

$$f_k = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{T} \mathbb{I}\{\mathrm{argmax}(p_{ij}) = k\}, \qquad (13)$$

where $p_{ij} = \mathrm{softmax}(W^T v_{ij} + b)$. Finally, we obtain the loss for training CITADEL:

$$\mathcal{L} = \mathcal{L}_{\mathrm{e}} + \mathcal{L}_{\mathrm{r}} + \alpha \cdot \mathcal{L}_{\mathrm{b}} + \beta \cdot \mathcal{L}_{\mathrm{s}}, \qquad (14)$$

where $\alpha \geq 0$ and $\beta \geq 0$. The ℓ_1 and load balancing loss are applied on both queries and documents.

3.5 Inverted Index Retrieval

CITADEL builds an inverted index like BM25 but we use a vector instead of a scalar for each token and the doc product as the term relevance.

Indexing and Post-hoc Pruning. To reduce index storage, we prune the vectors with routing weights less than a threshold τ after training. For a key e in the lexicon $\mathcal{V}$, the token index $\mathcal{I}_e$ consists of token embeddings v_{d_j} and the routing weight $w_{d_j}^e$ for all documents d in the corpus $\mathcal{C}$ is:

$$\mathcal{I}_e = \{w_{d_j}^e \cdot v_{d_j} \mid w_{d_j}^e > \tau, 1 \leq j \leq M, \forall d \in \mathcal{C}\}. \quad (15)$$

We will discuss the impact of posthoc pruning in Section 5.2, where we find that posthoc pruning can reduce the index size by $3\times$ without significant accuracy loss. The final search index is defined as $\mathcal{I} = \{\mathcal{I}_e | e \in \mathcal{V}\}$, where the load-balancing loss in Equation (11) will encourage the size distribution over $\mathcal{I}_e$ to be as even as possible. In practice, we set the number of maximal routing keys of each token to 5 for the document and 1 for the query. The intuition is that the documents usually contain more information and need more key capacity, which is discussed in Section 7 in detail.

Token Retrieval. Given a query q, CITADEL first encodes it into a sequence of token vectors $\{v_{q_i}\}_{i=1}^{N}$, and then route each vector to its top-1 key e with a routing weight $w_{q_i}^e$. The final representation $w_{q_i}^e \cdot v_{q_i}$ will be sent to the corresponding token index $\mathcal{I}_E$ for vector search. The final ranking list will be merged from each query token's document ranking according to Equation (5).

4 Experiments

4.1 MS MARCO Passages Retrieval

We evaluate MS MARCO passages (Nguyen et al., 2016) and its shared tasks, TREC DL 2019/2020 passage ranking tasks (Craswell et al., 2020). Dataset details are provided in Appendix A.1. Following standard practice, we train CITADEL and other baseline models on MS MARCO passages and report the results on its dev-small set and TREC DL 2019/2020 test queries. The evaluation metrics are MRR@10, nDCG@10, and Recall@1000 (i.e., R@1K). We provide a detailed implementation of CITADEL and other baselines in Appendix A.

Table 1 shows the in-domain evaluation results on MS MARCO passage and TREC DL 2019/2020. We divide the models into two classes: ones trained with only labels and BM25 hard negatives and the others trained with further pretraining (Gao and Callan, 2022), hard negative mining (Xiong et al., 2021), or distillation from a cross-encoder[1]. CITADEL is trained only with BM25 hard negatives, while CITADEL$^+$ is trained with cross-encoder distillation and one-round hard negative mining. The default pruning threshold is $\tau = 0.9$. As shown in Section 5.2, τ can be adjusted to strike different balances between latency, index size and accuracy. In both categories, CITADEL/CITADEL$^+$ outperforms the baseline models on the MS MARCO passages dev set and greatly reduces the search latency on both GPU and CPU. For example, CITADEL$^+$ achieves an average latency of 3.21 ms/query which is close to DPR-768 (1.28 ms/query) on GPU, while having a 25% higher MRR@10 score. CITADEL also maintains acceptable index sizes on disk, which can be further reduced using product quantization (Section 5.3). Although not able to outperform several baselines on TREC DL 2019/2020, we perform t-test (p < 0.05) on CITADEL and CITADEL$^+$ against other baselines in their sub-categories and

---

[1] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2

Models	MARCO Dev		TREC DL19		TREC DL20		Index Storage		Latency (ms/query)		
	MRR@10	R@1k	nDCG@10	R@1k	nDCG@10	R@1k	Disk (GB)	Factor[1]	Encode (GPU)	Search (GPU)	Search (CPU)
Models trained with only BM25 hard negatives											
BM25	0.188	0.858	0.506	0.739	0.488	0.733	**0.67**	**×0.22**	×	×	**40.1**
DPR-128	0.285	0.937	0.576	0.702	0.603	0.757	4.33	×1.42	7.09	**0.63**	430
DPR-768	0.319	0.941	0.611	0.742	0.591	0.796	26.0	×8.52	7.01	1.28	2015
SPLADE	0.340	0.965	0.683	0.813	0.671	0.823	2.60	×0.85	7.13	×	475
COIL-tok	0.350	0.964	0.660	0.809	0.679	0.825	52.5	×17.2	10.7	46.8	1295
COIL-full	0.353	0.967	0.704	0.835	0.688	0.841	78.5	×25.7	10.8	47.9	3258
ColBERT	0.360	0.968	0.694	0.830	0.676	0.837	154	×50.5	10.9	178	-
CITADEL	0.362	0.975	0.687	0.829	0.661	0.830	78.3	×25.7	10.8	3.95	520
Models trained with further pre-training/hard-negative mining/distillation											
coCondenser	0.382	0.984	0.674	0.820	0.684	0.839	26.0	×8.52	7.01	1.28	2015
SPLADE-v2	0.368	0.979	0.729	0.865	0.718	0.890	4.12	×1.35	7.13	×	2710
ColBERT-v2	0.397	**0.985**	**0.744**	**0.882**	**0.750**	**0.894**	29.0	×9.51	10.9	122	3275
ColBERT-PLAID[2]	0.397	0.984	0.744	0.882	0.749	0.894	22.1	×7.25	10.9	55.0	370
CITADEL+	**0.399**	0.981	0.703	0.830	0.702	0.859	81.3	×26.7	10.8	3.21	635

[1] Factor: Ratio of index size to plain text size.
[2] The PLAID implementation of ColBERT contains complex engineering schemes and low-level optimization such as centroid interaction and fast kernels.

Table 1: In-domain evaluation on MS MARCO passages and TREC DL 2019/2020. CITADEL+ is trained with cross-encoder distillation and hard-negative mining. The red region means CITADEL/CITADEL+ is better than the method while the green region means that there's no statistical significance ($p > 0.05$). "×" means not implemented and "-" means not practical to evaluate on a single CPU.

Methods	AA	CF	DB	Fe	FQ	HQ	NF	NQ	Qu	SF	SD	TC	T2	Avg.
Models trained with only BM25 hard negatives														
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
SPLADE	0.445	0.201	0.370	0.740	0.289	0.640	0.322	0.469	0.834	0.633	0.149	0.661	0.201	0.453
COIL-full	0.295	0.216	0.398	**0.840**	0.313	**0.713**	0.331	0.519	0.838	**0.707**	0.155	0.668	0.281	0.483
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
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
Models with further pre-training/hard-negative mining/distillation														
coCondenser	0.440	0.133	0.347	0.511	0.281	0.533	0.319	0.467	0.863	0.591	0.130	0.708	0.143	0.420
SPLADE-v2	0.479	**0.235**	0.435	0.786	0.336	0.684	0.334	0.521	0.838	0.693	0.158	0.710	0.272	0.499
ColBERT-v2	0.463	0.176	**0.446**	0.785	**0.356**	0.667	**0.338**	**0.562**	**0.854**	0.693	**0.165**	**0.738**	0.263	0.500
CITADEL+	0.490	0.181	0.420	0.747	0.332	0.652	0.337	0.539	0.852	0.695	0.147	0.680	0.340	0.493
CITADEL+ (w/o reg.)	**0.511**	0.182	0.422	0.765	0.330	0.664	0.337	0.540	0.853	0.690	0.159	0.715	0.342	**0.501**

Table 2: Out-of-domain evaluation on BEIR benchmark. nDCG@10 score is reported. Dataset Legend (Chen et al., 2022): TC=TREC-COVID, NF=NFCorpus, NQ=NaturalQuestions, HQ=HotpotQA, FQ=FiQA, AA=ArguAna, T2=Touché-2020 (v2), Qu=Quora, DB=DBPedia, SD=SCIDOCS, Fe=FEVER, CF=Climate-FEVER, SF=SciFact.

show there is no statistical significance. The inconsistency is probably due to that we use the training data from Tevatron (Gao et al., 2022) where each passage is paired with a title. Lassance and Clinchant (2023) points out that neural retrievers trained on such data will result in slightly higher scores on MS MARCO dev small while lower scores on TREC DL 2019 and 2020.

4.2 BEIR: Out-of-Domain Evaluation

We evaluate on BEIR benchmark (Thakur et al., 2021) which consists of a diverse set of 18 retrieval tasks across 9 domains. We evaluate on 13 datasets following previous works (Santhanam et al., 2022b; Formal et al., 2021a). Table 2 shows the zero-shot evaluation results on BEIR. Without

any pre-training or distillation, CITADEL manages to outperform all baselines in their sub-categories in terms of the average score. Compared with the distilled/pre-trained models, CITADEL+ still manages to achieve comparable performance. Interestingly, we find that if no regularization like load balancing and L1 is applied during training, CITADEL+ can reach a much higher average score that even outperforms ColBERT-v2. Our conjecture is that the regularization reduces the number of token interactions and the importance of such token interaction is learned from training data. It is hence not surprising that the more aggressively we prune token interaction, the more likely that it would hurt out-of-domain accuracy that's not covered by the training data.

Models	MRR@10	#DP $\times 10^6$
ColBERT	0.360	4213
COIL-full	0.353	45.6
CITADEL	0.362	10.5
DPR-128	0.285	8.84

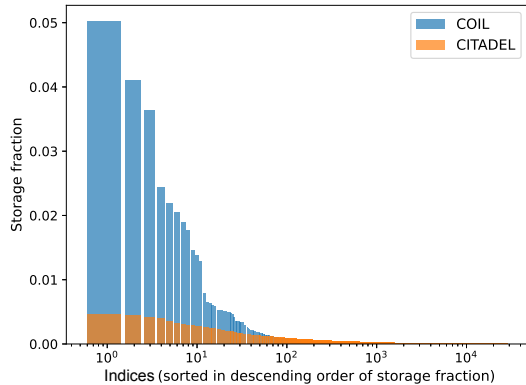Table 3: Maximal number of dot products per query on MS MARCO passages. # DP: number of dot products.

Figure 3: Normalized disk storage over token indices in an inverted list of CITADEL and COIL.

Figure 4: Latency-memory-accuracy tradeoff on MS MARCO passages using post-hoc pruning. τ is the pruning threshold.

Condition	MRR@10	Storage (GB)	Latency (ms)
original	0.362	78.3	3.95
nbits=2	0.361	13.3	0.90
nbits=1	0.356	11.0	0.92

Table 4: Product Quantization. Pruning threshold is set to 0.9. nbits: ratio of total bytes to the vector dimension.

5 Performance Analysis

5.1 Number of Token Interactions

The actual latency is often impacted by engineering details and therefore FLOPS is often considered for comparing efficiency agnostic to the actual implementation. In our case, however, FLOPS is impacted by the vector dimension in the nearest neighbour search which is different across models. Therefore, we only compare the maximal number of dot products needed as a proxy for token interaction per query during retrieval as shown in Table 3. The number of dot products per query in CITADEL with pruning threshold $\tau = 0.9$ is comparable to DPR-128 and much lower than ColBERT and COIL, which is consistent with the latency numbers in Table 1. The reason is that CITADEL has a balanced inverted index credited to the ℓ_1 regularization and the load balancing loss as shown in Figure 3. By applying the load balancing loss on the router prediction, CITADEL yields a more balanced and even index distribution where its largest index fraction is 8 × smaller than COIL's as shown in Figure 3. We also provide a detailed latency breakdown in Appendix A.4.
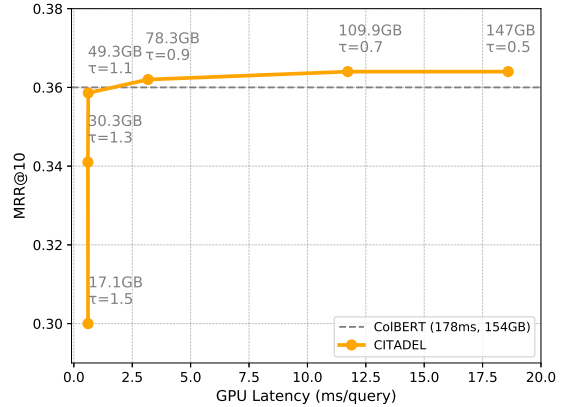
5.2 Latency-Memory-Accuracy Tradeoff

Figure 4 shows the tradeoff among latency, memory, and MRR@10 on MS MARCO passages with post-hoc pruning. We try the pruning thresholds [0.5, 0.7, 0.9, 1.1, 1.3, 1.5]. We could see that the MRR@10 score barely decreases when we increase the threshold to from 0.5 to 1.1, but the latency decreases by a large margin, from about 18 ms/query to 0.61 ms/query. The sweet spots are around (0.359 MRR@10, 49.3GB, 0.61 ms/query) and (0.362 MRR@10, 78.5GB, 3.95 ms/query). This simple pruning strategy is extremely effective and readers can see in Section 6 that it also yields interpretable document representations.

5.3 Combination with Product Quantization

We could further reduce the latency and storage with product quantization (Jégou et al., 2011) (PQ) as shown in Table 4. For nbits=2, we divide the vectors into sets of 4-dimensional sub-vectors and use 256 centroids for clustering the sub-vectors, while for nbits=1 we set the sub-vector dim to 8 and the same for the rest. With only 2 bits per dimension, the MRR@10 score on MS MARCO Dev only drops 4% but the storage is reduced by 83% and latency is reduced by 76%.

Threshold τ	Sample documents from MS MARCO Passages
0.0	**All medications have side effects**, including **drugs** to **treat arrhythmias**. Most of the **side effects** aren't **serious** and **disappear** **when** the dose is **changed** or the **medication** is **stopped**. But some **side effects** are very **serious**. That's why some **children** are **admitted** to the **hospital** to begin the **medication**. **Medications for Arrhythmia**
0.9	All **medications** have **side effects**, including **drugs** to treat **arrhythmias**. Most of the **side effects** aren't **serious** and **disappear** when the dose is changed or the **medication** is **stopped**. But some **side effects** are very **serious**. That's why some children are **admitted** to the **hospital** to begin the **medication**. **Medications** for **Arrhythmia**
1.3	All medications have side effects, including drugs to treat **arrhythmias**. Most of the side effects aren't serious and disappear when the dose is changed or the medication is stopped. But some side effects are very serious. That's why some children are admitted to the hospital to begin the medication. Medications for **Arrhythmia**

Figure 5: Tokens in grey (pruned) have zero activated keys while bold tokens have at least one activated key. We leave out the expanded terms and routing weights due to space limit.

Models	Dev	DL19	Latency (ms)
COIL-full	0.353	0.704	47.9
COIL-tok	0.350	0.660	46.8
CITADEL	0.362	0.687	3.95
CITADEL-tok	0.360	0.665	1.64

Table 5: `[CLS]` ablation on MS MARCO passage. Dev: MRR@10. DL19: nDCG@10.

#Keys	MRR@10	Storage (GB)	Latency (ms)
1	0.347	53.6	1.28
3	0.360	136	14.7
5	0.364	185	18.6
7	0.370	196	20.4
9	0.367	221	19.6

Table 6: Number of routing keys for documents during training. No post-hoc pruning is applied.

6 Token Routing Analysis of CITADEL

Qualitative Analysis. We visualize CITADEL representations and the effect of posthoc pruning in Figure 5. By increasing the pruning threshold, more keywords are pruned and finally leaving the most central word "*arrhythmia*" activated. We provide another example in Figure 6. We can see that a lot of redundant words that do not contribute to the final semantics are deactivated, meaning all their routing weights are 0. For the activated tokens, we could see the routed keys are contextualized as many of them are related to *emoji* which is the theme of the document.

Quantitative Analysis. We analyze CITADEL's token distribution over the number of activated routing keys for the whole corpus as shown in Figure 7. With ℓ_1 loss, around 50 tokens per passage are deactivated (i.e., all the routing weights of these 50 tokens are 0). As the pruning threshold increases, more tokens are deactivated, yielding a sparse representation for interpreting CITADEL's behaviours.

7 Ablation Studies

Impact of `[CLS]` Table 5 shows the influence of removing the `[CLS]` vector for CITADEL on MS MARCO passage. Although removing `[CLS]` improves the latency by a large margin, the in-domain effectiveness is also adversely affected, especially

on TREC DL 2019. Nevertheless, CITADEL-tok (w/o `[CLS]`) still outperforms its counterpart COIL-tok in both precision and latency.

Number of Routed Experts. Table 6 shows the influence of changing the maximum number of keys that each document token can be routed to during training and inference on MS MARCO passage. As the number of routing keys increases, the index storage also increases rapidly but so does the MRR@10 score which plateaus after reaching 7 keys. The latency does not increase as much after 3 routing keys due to the load balancing loss.

8 Related Works

Dense Retrieval. Supported by multiple approximate nearest neighbour search libraries (Johnson et al., 2021; Guo et al., 2020), dense retrieval (Karpukhin et al., 2020) gained much popularity due to its efficiency and flexibility. To improve effectiveness, techniques such as hard negative mining (Xiong et al., 2021; Zhan et al., 2021) and knowledge distillation (Lin et al., 2021b; Hofstätter et al., 2021) are often deployed. Recently, retrieval-oriented pre-training(Gao et al., 2021b; Lu et al., 2021; Gao and Callan, 2021; Izacard et al., 2022; Gao and Callan, 2022) also draws much attention as they could substantially improve the fine-tuning performance of downstream tasks.

Sample Document from MS MARCO Passages
It may be a little hard to **remember** (remember, forget) all of the **25** (barney, many, 25, 1625) **codes** (code, codes), which is why we've **created** (invented, stumbled, created, swirled) this **dictionary** (dictionary, spiked). You can find the complete **facebook** (facebook) **emoticons** (emoticon, emoji, combinations, asher, grins) **list** for **chats** (chat, chatting) **and status** (winked, logo, status, glided) in the table above. This **list** (list, kingsley, include)ing includes all the new ones which have been recently added, such as the **penguin** (penguin, mccann, uefa) or **shark** (shark, chao) **emoticon** (emoticon, emoji, assassins). **Facebook** (facebook) **Emoticons** (emoticon, emoji, spp, programs)

Figure 6: Token routing analysis of CITADEL. Grey tokens are deactivated, while bold tokens are routed to at least one activated key (in blue).

Figure 7: Token distribution over number of activated experts per passage. τ is the pruning threshold.

Sparse Retrieval. Traditional sparse retrieval systems such as BM25 (Robertson and Zaragoza, 2009) and tf–idf (Salton and Buckley, 1988) represent the documents as a bag of words with static term weights. Recently, many works leverage pre-trained language models to learn contextualized term importance (Bai et al., 2020; Mallia et al., 2021; Formal et al., 2021b; Lin and Ma, 2021). These models could utilize existing inverted index libraries such as Pyserini (Lin et al., 2021a) to perform efficient sparse retrieval or even hybrid with dense retrieval (Hofstätter et al., 2022; Shen et al., 2022; Lin and Lin, 2022; Zhang et al., 2023).

Multi-Vector Retrieval. ColBERT (Khattab and Zaharia, 2020; Santhanam et al., 2022b,a; Hofstätter et al., 2022) probably has the most optimized library in multi-vector retrieval. COIL (Gao et al., 2021a) accelerates retrieval by combining with exact match and inverted vector search. ME-BERT (Luan et al., 2021) and MVR (Zhang et al., 2022) propose to use a fixed number of token embeddings for late interaction (e.g., top-k positions or special tokens). Concurrently to this work, ALIGNER (Qian et al., 2022) proposes to frame multi-vector retrieval as a sparse alignment problem between query tokens and document tokens using entropy-regularized linear programming. Our 110M model achieves higher in-domain and out-of-domain accuracy than their large variants.

9 Conclusion

This paper proposes a novel multi-vector retrieval method that achieves state-of-the-art performance on several benchmark datasets while being 40× faster than ColBERT-v2 and 17× faster than the most efficien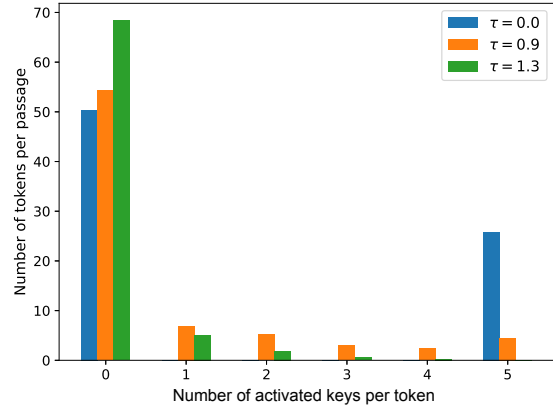t multi-vector retrieval library to date, PLAID, on GPUs. By jointly optimizing for the token index size and load balancing, our new dynamic lexical routing scheme greatly reduces the redundancy in the all-to-all token interaction of Col-BERT while bridging the word-mismatch problem in COIL. Experiments on both in-domain and out-of-domain datasets demonstrate the effectiveness and efficiency of our model.

10 Limitations

The limitation of CITADEL mainly shows in two aspects. First, at the beginning of training, the model needs to route each token vector to multiple activated keys for token interaction, which increases the computation cost compared to COIL and Col-BERT. This results in slower training speed but it gets better when training approaches the end as more tokens are pruned by the ℓ_1 regularization. Another drawback lies in the implementation of CITADEL, or more generally speaking, most multi-vector retrieval methods. The token-level retrieval and aggregation make them not compatible with established search libraries such as FAISS or Pyserini. Moreover, for time and space efficiency, multi-vector retrieval also requires more engineering efforts and low-level optimization. Recently, XTR (Lee et al., 2023) provides a solution that constrains the document-level retrieval to be consistent with the token-level retrieval during training, which can be used for streamlining CITADEL.

11 Acknowledgement

References

Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. Sparterm: Learning term-based sparse representation for fast text retrieval. *ArXiv*, abs/2010.00768.

Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2020. Overview of touché 2020: Argument retrieval: Extended abstract. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings*, page 384–395, Berlin, Heidelberg. Springer-Verlag.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval*, pages 716–722, Cham. Springer International Publishing.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Xilun Chen, Kushal Lakhotia, Barlas Oğuz, Anchit Gupta, Patrick Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen-tau Yih. 2022. Salient phrase aware dense retrieval: Can a dense retriever imitate a sparse one? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. Specter: Document-level representation learning using citation-informed transformers.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. 2020. Overview of the trec 2020 deep learning track. *ArXiv*, abs/2003.07820.

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.

Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2021. Climate-fever: A dataset for verification of real-world climate claims.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Thibault Formal, C. Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021a. Splade v2: Sparse lexical and expansion model for information retrieval. *ArXiv*, abs/2109.10086.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021b. 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*. Association for Computing Machinery.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 981–993, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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. 2021a. 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. 2022. Tevatron: An efficient and flexible toolkit for dense retrieval.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896. PMLR.

Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. Dbpedia-entity v2: A test

collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 1265–1268, New York, NY, USA. Association for Computing Machinery.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Sebastian Hofstätter, Omar Khattab, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. Introducing neural bag of whole-words with colberter: Contextualized late interactions using enhanced reduction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, page 737–747. Association for Computing Machinery.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 113–122. Association for Computing Machinery.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128.

J. Johnson, M. Douze, and H. Jegou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(03):535–547.

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 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 39–48. Association for Computing Machinery.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering

research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Carlos Lassance and Stéphane Clinchant. 2023. The tale of two ms marco – and their unfair comparisons.

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.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021a. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 2356–2362. Association for Computing Machinery.

Jimmy J. Lin and Xueguang Ma. 2021. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. *ArXiv*, abs/2106.14807.

Sheng-Chieh Lin and Jimmy Lin. 2022. A dense representation framework for lexical and semantic matching. *ArXiv*, abs/2206.09912.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021b. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net.

Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. 2021. Less is more: Pretrain a strong Siamese encoder for dense text retrieval using a weak decoder. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2780–2791, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Joel Mackenzie, Andrew Trotman, and Jimmy Lin. 2021. Wacky weights in learned sparse representations and the revenge of score-at-a-time query evaluation. *ArXiv*, abs/2110.11540.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1941–1942, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning passage impacts for inverted indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1723–1727. Association for Computing Machinery.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with limoe: the language-image mixture of experts. *ArXiv*, abs/2206.02770.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

Natasha Noy, Matthew Burgess, and Dan Brickley. 2019. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *28th Web Conference (WebConf 2019)*.

Yujie Qian, Jinhyuk Lee, Sai Meher Karthik Duddu, Zhuyun Dai, Siddhartha Brahma, Iftekhar Naim, Tao Lei, and Vincent Zhao. 2022. Multi-vector retrieval as sparse alignment. *ArXiv*, abs/2211.01267.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. Plaid: An efficient engine for late interaction retrieval. *ArXiv*, abs/2205.09707.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. 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.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Kai Zhang, and Daxin Jiang. 2022. Unifier: A unified retriever for large-scale retrieval. *ArXiv*, abs/2205.11194.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. Trec-covid: Constructing a pandemic information retrieval test collection.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251, Melbourne, Australia. Association for Computational Linguistics.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online. Association for Computational Linguistics.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and

Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1503–1512. Association for Computing Machinery.

Kai Zhang, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, Binxing Jiao, and Daxin Jiang. 2023. Led: Lexicon-enlightened dense retriever for large-scale retrieval. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 3203–3213, New York, NY, USA. Association for Computing Machinery.

Shunyu Zhang, Yaobo Liang, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Multi-view document representation learning for open-domain dense retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5990–6000, Dublin, Ireland. Association for Computational Linguistics.

A Implementations

A.1 Datasets

The MS MARCO passages corpus has around 8.8 million passages with an average length of 60 words. TREC DL 2019 and 2020 contain 43 and 54 test queries whose relevance sets are densely labelled with scores from 0 to 4.

For out-of-domain evaluation, we use 13 datasets from BEIR, which includes TREC-COVID (Voorhees et al., 2020), NFCorpus (Boteva et al., 2016), Natural Questions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), FiQA-2018 (Maia et al., 2018), ArguAna Counterargs Corpus (Wachsmuth et al., 2018), Touché-2020 (Bondarenko et al., 2020), Quora[2], DBPedia-Entity-v2 (Hasibi et al., 2017), SCIDOCS (Cohan et al., 2020), FEVER (Thorne et al., 2018), Climate-FEVER (Diggelmann et al., 2021), SciFact (Wadden et al., 2020).

A.2 Baselines for Section 4

All the baseline models below are trained and evaluated under the same setting of CITADEL (e.g., datasets, hyperparameters, and hardwares).

Sparse Retrievers. BM25 (Robertson and Zaragoza, 2009) uses the term frequency and inverted document frequency as features to compute the similarity between documents. SPLADE (Formal et al., 2021b,a) leverages the pre-trained language model's MLM layer and ReLU activation to yield sparse term importance.

Dense Retrievers. DPR (Karpukhin et al., 2020) encodes the input text into a single vector. coCondenser (Gao and Callan, 2022) pre-trains DPR in an unsupervised fashion before fine-tuning.

Multi-Vector Retrievers. ColBERT (Khattab and Zaharia, 2020; Santhanam et al., 2022b) encodes each token into dense vectors and performs late interaction between query token vectors and document token vectors. COIL (Gao et al., 2021a) applies an exact match constraint on late interaction to improve efficiency and robustness.

A.3 Training

For CITADEL, we use bert-base-uncased as the initial checkpoint for fine-tuning. Following COIL,
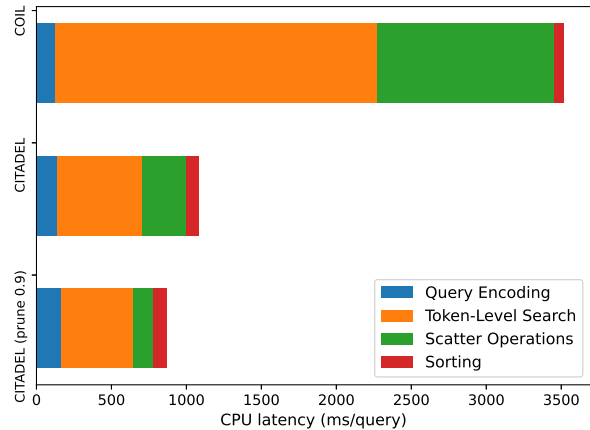

Figure 8: Latency breakdown of inverted vector retrieval for CITADEL and COIL.

we set the `[CLS]` vector dimension to 128, token vector dimension to 32, maximal routing keys to 5 for document and 1 for query, α and β in Equation (14) are set to be 1e-2 and 1e-5, respectively. We add the dot product of `[CLS]` vectors in Equation (1) to the final similarity score in Equation (5). All models are trained for 10 epochs with AdamW (Loshchilov and Hutter, 2019) optimizer, a learning rate of 2e-5 with 3000 warm-up steps and linear decay. Hard negatives are sampled from top-100 BM25 retrieval results. Each query is paired with 1 positive and 7 hard negatives for faster convergence. We use a batch size of 128 on MS MARCO passages with 32 A100 GPUs.

For a fair comparison with recent state-of-the-art models, we further train CITADEL using cross-encoder distillation and hard negative mining. First, we use the trained CITADEL model under the setting in the last paragraph to retrieve top-100 candidates from the corpus for the training queries. We then use the cross-encoder[3] to rerank the top-100 candidates and score each query-document pair. Finally, we re-initialize CITADEL with bert-base-uncased using the positives and negatives sample from the top-100 candidates scored by the cross-encoder, with a 1:1 ratio for the soft-label and hard-label loss mixing (Hinton et al., 2015). We also repeat another round of hard negative mining and distillation but it does not seem to improve the performance any further.

A.4 Inference and Latency Breakdown

Pipeline. We implemented the retrieval pipeline with PyTorch (GPU) and Numpy (CPU), with a small Cython extension module for scatter operations similar to COIL's[4]. As shown in Fig 8, our pipeline could be roughly decomposed into four independent parts: query encoding, token-level retrieval, scatter operations, and sorting. We use the same pipeline for COIL's retrieval process. For ColBERT's latency breakdown please refer to Santhanam et al. (2022a). The cost of query encoding comes from the forward pass of the query encoder, which could be independently optimized using quantization or weight pruning for neural networks. Besides that, the most expensive operation is the token-level retrieval, which is directly influenced by the token index size. We could see that a more balanced index size distribution as shown in Figure 3 has a much smaller token vector retrieval latency. The scatter operations are used to gather the token vectors from the same passage ids from different token indices, which is also related to the token index size distribution. Finally, we sort the aggregated ranking results and return the candidates.

Hardwares and Latency Measurement. We measure all the retrieval models in Table 1 on a single A100 GPU for GPU search and a single Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz for CPU search. All indices are stored in fp32 (token vectors) and int64 (corpus ids if necessary) on disk. We use a query batch size of 1 and return the top-1000 candidates by default to simulate streaming queries. We compute the average latency of all queries on MS MARCO passages' Dev set and then report the minimum average latency across 3 trials following PLAID (Santhanam et al., 2022a). I/O time is excluded from the latency but the time of moving tensors from CPU to GPU during GPU retrieval is included.

---

[4] https://github.com/luyug/COIL/tree/main/retriever

A For every submission:

☑ A1. Did you describe the limitations of your work?
10

☒ A2. Did you discuss any potential risks of your work?
This work provides an information retrieval method for public datasets.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

4, 5, Appendix A

☑ B1. Did you cite the creators of artifacts you used?
4, 5, Appendix A

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
4, 5, Appendix A

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
4, 5, Appendix A

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The public datasets in the paper are widely used for a long time.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☑ Did you run computational experiments?

4,5,6,7

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix A

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix A

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.