Cluster-based Partial Dense Retrieval Fused with Sparse Text Retrieval

Yingrui Yang
Department of Computer Science,
University of California
Santa Barbara, California, USA

Parker Carlson
Department of Computer Science,
University of California
Santa Barbara, California, USA

Shanxiu He
Department of Computer Science,
University of California
Santa Barbara, California, USA

Yifan Qiao
Department of Computer Science,
University of California
Santa Barbara, California, USA

Tao Yang
Department of Computer Science,
University of California
Santa Barbara, California, USA

ABSTRACT

Previous work has demonstrated the potential to combine document rankings from dense and sparse retrievers for higher relevance effectiveness. This paper proposes a cluster-based partial dense retrieval scheme guided by sparse retrieval results to optimize fusion between dense and sparse retrieval at a low space and CPU-time cost while retaining a competitive relevance. This scheme exploits the overlap of sparse retrieval results and document embedding clusters, and judiciously selects a limited number of clusters to probabilistically guarantee the inclusion of top sparse results. This paper provides an evaluation of this scheme on its in-domain and zero-shot retrieval performance for the MS MARCO and BEIR datasets.

CCS CONCEPTS

• **Information systems → Combination, fusion and federated search**.

KEYWORDS

Space and time efficient search, dense retrieval, sparse retrieval.

ACM Reference Format:
Yingrui Yang, Parker Carlson, Shanxiu He, Yifan Qiao, and Tao Yang. 2024. Cluster-based Partial Dense Retrieval Fused with Sparse Text Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626772.3657972

1 INTRODUCTION AND RELATED WORK

Recent studies have found that combining sparse and dense retrieval scores can further boost retrieval relevance [15, 18, 19], suggesting that two categories of retrievers capture different types of relevant signals. There exists a resource requirement imbalance between the computing platforms to run these two retrievers: 1) Sparse retrieval, utilizing a compact inverted index, operates efficiently on a low-cost CPU server, especially with the recent retriever efficiency optimization for learned representations [7, 8, 16, 17, 19, 24, 25, 27, 28, 31, 32, 34]; 2) Dense retrieval requires GPU support due to the time-consuming nature of similarity calculation in scanning through a large dataset. The increase in the data size and/or embedding dimensionality, particularly in large language models [1, 37] (e.g. from 768 to 12,288), demands substantially more memory and computing time. For example, RepLLaMA retrieval [22], based on LLaMA-2 [37], requires 145GB storage space for 8.8M MARCO passage embeddings. In general, dense retrieval is a resource-intensive bottleneck in the fusion of dense and sparse retrieval systems. Running two retrievers independently on a search platform and use naive interpolation may overlook potential opportunities.

This paper focuses on reducing the resource requirements of the above fusion through improved partial dense retrieval, to produce competitive ranking results at a low space and CPU time cost. Previous work has explored two categories of partial dense retrieval within a portion of the corpus. The first category employs unsupervised clustering to group documents offline [12], followed by the online selection of top clusters using query-centroid distances. Such a method is termed as Inverted File Index (IVF) cluster search. As shown in Section 3, there can be a significant relevance drop using limited IVF cluster search. The other category takes a graph navigation approach such as HNSW [26], GAR [23] and LADR [14]. They start from an initial seed or a ranked sparse result list and select a subset of embeddings based on document-to-document similarity-based proximity links.

In addition to partial dense retrieval, quantization methods [39, 41, 42] such as PQ and OPQ are orthogonal strategies employed for memory and computing cost reduction in above approaches. However, they come with a tradeoff in relevance effectiveness, as observed in [39]. For example, Section 3 shows up-to 20% drops of MRR@10 in MS MACRO passage dense retrieval using OPQ with RetroMAE [40]. For applications with a stringent relevance requirement, utilizing uncompressed dense embeddings is still preferred, even though they may not fit into memory.

We opt to take the cluster-based approach for partial dense retrieval instead of graph navigation, inspired by two observations: 1) Maintaining a document-to-document proximity graph is expensive for a large corpus. For instance, in LADR [14], sustaining the proximity graph for 8.8M passages in the MS MARCO dataset takes an extra 4.3GB memory space, in contrast to the original 27GB dense

SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0431-4/24/07.
https://doi.org/10.1145/3626772.3657972

embeddings and <1GB index after quantization. 2) Embedding vectors of a large dataset and their proximity graph, especially with high-dimensional vectors, may exceed memory capacity and require dynamic access to disk storage. There are optimization studies on graph-based nearest neighbor search through disk data [10, 11]. Nevertheless, random access of embeddings and/or graph nodes, as necessitated by document-level proximity graph navigation, can incur substantial fine-grained I/O overhead. In contrast, the cluster-based approach, which captures similar documents within each statistically-formed group, enables faster block-level I/O.

Previous studies on IVF cluster in-memory or on-disk search [3, 12] have not considered the fusion with sparse retrieval, and its partial dense retrieval is purely guided by query-centroid similarity. There exists an opportunity to exploit result relationship between sparse and dense retrievers to enhance relevance effectiveness and efficiency. To this end, we propose a Cluster-based partial Dense retrieval scheme Fused with Sparse retrieval (**CDFS** for short).

2 PROPOSED CDFS SCHEME

Problem definition. Given query q and a collection of D text documents: $\{d_i\}_{i=1}^{D}$, retrieval obtains the top k relevant documents based on the similarity between the representation vectors of q and $\{d_i\}_{i=1}^{D}$. Sparse retrieval uses the dot product of their lexical representation vectors as the similarity function: $L(q) \cdot L(d_i)$ where $L(.)$ is a vector of weighted term tokens for a document or a query [5, 8, 19, 27, 33]. Its implementation uses an inverted index for efficient search. Dense retrieval with a single vector representation computes the following rank score $R(q) \cdot R(d_i)$ where $R(.)$ is a dense representation vector of a fixed size [13]. Following the work of [9, 15, 18], linear interpolation is used to ensemble dense and sparse retrieval scores. The fused rank score of document d_i for query q is: $\lambda L(q) \cdot L(d_i) + (1-\lambda) R(q) \cdot R(d_i)$, where λ is a coefficient between 0 and 1, and the sparse and dense rank scores are normalized. Two design considerations in CDFS are:

- **Re-ranking of top sparse retrieval results and their similar documents.** Re-ranking of top sparse results is a simple strategy to guarantee the inclusion of these top results. However, as pointed out in the previous work [14, 23], re-ranking does not improve recall. We extend this idea to select the clusters that include top sparse results. Thus, both top sparse results and documents closely similar to the top results are evaluated by CDFS.
- **Exploiting membership distribution of top sparse retrieval results**. Given the cluster information of dense representations, we examine the membership of sparse retrieval results in these clusters in a weighted manner, so that clusters that contain multiple top sparse results have a higher priority to be evaluated. This weighted scheme allows CDFS to identify more diverse candidates similar to top sparse results to improve the recall.

During offline processing, documents are clustered using k-means following the standard practice [12, 21]. For collections with a large number of high-dimension embeddings, which may not fit into memory, we employ a k-means variant called bisecting k-means [35] to recursively divide a large cluster into smaller clusters.

The online query processing flow of the proposed CDFS scheme during retrieval is shown in Figure 1. Initially, sparse retrieval produces top-k results. Then, CDFS conducts the following steps

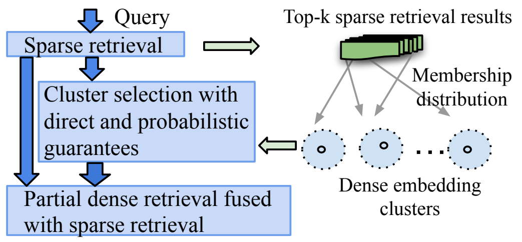
Query
Sparse retrieval
Top-k sparse retrieval results
Cluster selection with direct and probabilistic guarantees
Membership distribution
Dense embedding clusters
Partial dense retrieval fused with sparse retrieval

Figure 1: Flow of CDFS for selective retrieval fusion

to select a subset of clusters as G_k and fuse. Three parameters used are α, β, and γ, satisfying $\alpha \le \gamma$ and $\beta \le \gamma$.

(1) Assign a priority score $W(C_i)$ to each embedding cluster C_i using the weighted overlap of top k sparse results with C_i:

$$W(C_i) = \sum_{d_j \in C_i \cap TopS(k)} \frac{SparseScore(d_j)}{\log(RankS(d_j)+1)} \qquad (1)$$

where $SparseScore(d_j)$ is the sparse rank score of document d_j, $TopS(k)$ is the set of top-k sparse retrieval results, and $RankS(d_j)$ is the rank position of document d_j in $TopS(k)$.

(2) Include any cluster C_i in the targeted set G_k if
   - C_i contains any top-αk sparse result,
   - or $W(C_i) \ge \Theta_{\beta k}$.

   Threshold $\Theta_{\beta k}$ is determined below. It guarantees top-βk sparse results are included with approximate probability $1-\epsilon$.

(3) To limit the size of Set G_k, we further trim it by first keeping clusters of G_k which contain top-max$(\alpha, \beta)k$ sparse results, and then retaining additional top-weighted clusters under formula $W(C_i)$ as long as $|G_k| \le \gamma k$.

Compared to direct top-αk guarantee, the above probabilistic guarantee using overlap-weighted $W(C_i)$ identifies clusters with more diverse results. This is because a cluster that hosts multiple top sparse results may gain a higher priority score $W(C_i)$, compared to a cluster that hosts one top sparse result, even though its weighting is scaled by its rank position.

Complexity. The time complexity of partial dense retrieval in CDFS is $O(\gamma k * C * Dim)$. Here, C is the average cluster size and Dim is the embedding dimensionality. Our evaluation with $k = 1000$ uses $\gamma = 0.06$ for BM25 and 0.03 for SPLADE. Extra space overhead is $O(D)$ for mapping D documents to their cluster IDs, which is negligible compared to the dense embedding space cost.

Threshold for a probabilistic guarantee. We compute threshold $\Theta_{\beta k}$ by estimating the minimum score of a cluster that approximately includes top-βk sparse result based on order statistics [38]. As discussed in [30, 38], order statistics asymptotically follow a normal distribution. As sparse retrieval search is conducted on the whole document corpus, this asymptotic property can be applied to its top k-th result for each query, assuming it is an independent and identically distributed random variable. Let $S_{(k)}$ be the k-th rank score by sparse retrieval for a query, and based on the order statistics, $S_{(k)}$ converges in distribution to a normal distribution:

$$S_{(k)} \xrightarrow{d} N(\mu_k, \sigma_k^2).$$

A sample of queries can be used to compute the unbiased estimator of μ_k and σ_k^2. The steps for threshold $\Theta_{\beta k}$ estimation are as follows.

(1) Given m sampled queries processed using sparse retrieval, for each position k, under the assumption of normality, we derive an unbiased estimation of the mean and variance of this normal distribution:

$$\hat{\mu_k} = \frac{1}{m}\sum_q S_{(k),q}, \quad \hat{\sigma_k} = \sqrt{\frac{\sum_q (S_{(k),q} - \hat{\mu_k})^2}{m}},$$

where q is one of the m sampled queries and $S_{(k),q}$ is the k-th rank score for query q.

(2) As the $S_{(k)}$ converges in distribution to a normal distribution,

$$\lim_{m\to\infty} Pr(\frac{S_{(k)} - \mu_k}{\sigma_k} \leq Z_\epsilon) = \epsilon, \text{ and } \quad \epsilon = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{Z_\epsilon} e^{0.5x^2} dx.$$

Thus given ϵ, one can find Z_ϵ following a standard statistical solution to the second equation expression above.

Notice that $S_{(k)} \geq \phi_k$ with at least probability $1 - \epsilon$, where $\phi_k = \mu_k + Z_\epsilon \sigma_k$. We can estimate ϕ_k with $\hat{\phi_k} = \hat{\mu_k} + Z_\epsilon \hat{\sigma_k}$.

(3) From Formula (1), we compute the lower bound of $W(C_i)$ when it contains at least one top-βk document. Namely,

$$W(C_i) \geq \frac{\phi_{\beta k}}{log(\beta k + 1)}.$$

We will use $\Theta_{\beta k} = \frac{\hat{\Phi}_{\beta k}}{log(\beta k+1)}$ as the threshold to add the clusters to $O_{\beta k}$, which satisfy $W(C_i) \geq \Theta_{\beta k}$ for Step (2) of cluster selection discussed earlier.

For example, in the case of SPLADE, the estimated mean is $\hat{\mu_{\beta k}} = 0.69$ and standard deviation $\hat{\sigma_{\beta k}} = 0.18$ for rank position $\beta k = 10$. With probability $1 - \epsilon = 0.95$, a random query will have its top-10 score above $\phi_{10} = 0.72$. Then, $\Theta_{10} = 0.30$. After setting $W(C_i)$ threshold to be 0.30 , the inclusion of a top-10 sparse result in each selected cluster will have a probability of 0.95.

3 EVALUATIONS

Datasets and measures. Our evaluation uses the MS MARCO dataset with 8.8 million passages for full passage ranking [2, 4]. The test query sets are the Dev set with 6980 queries and TREC deep learning (DL) 2019/2020 tracks with 43 and 54 queries each. For Dev, we report mean reciprocal rank at 10 (MRR@10) and recall at 1000 (R@1k). For DL19 and DL20 sets, we report NDCG@10. We report average NDCG@10 for the thirteen BEIR datasets [36].

Models and parameters. Retriever implementation is extended from PISA [29], FAISS, and LADR [14] in C++ and Python. For sparse and dense model interpolation, we use min-max normalization to rescale the top results per query. The interpolation weights are 0.05 and 0.95 for sparse and dense scores respectively when fused with BM25. For a learned sparse representation, they are 0.5 and 0.5, respectively. To report the mean single-query latency and 99 percentile latency in milliseconds, we run test queries multiple times for MS MARCO Dev set using a single thread on on an Intel i7-1260P CPU server with 64GB memory and PCIe SSD. The related code will be in https://github.com/yingrui-yang/dense_sparse_fusion. For sparse retrieval, our evaluation reported in this paper mainly uses a version of SPLADE [6, 7] with efficiency optimization [28, 31, 32]. Its index space is 3.7GB with CPU latency 18.5ms. Use of CDFS with other sparse retrievers [19, 34?] has similar findings.

For dense retrieval, we adopt two recently published dense models RetroMAE [40] based on BERT and RepLLaMA [22] based on LLaMA-2 [37]. We use their checkpoints from Huggingface to generate document and query embeddings and use k-means from the FAISS library [12] to derive 60,000 and 66,489 dense embedding clusters respectively for 8.8M MS MARCO passages. On average the cluster size is 145 and 133 documents for the two models. For the BEIR datasets, we set the desired cluster size to be 133 for each dataset and vary the number of clusters accordingly. The RetroMAE embedding set takes 27GB of space with a flax index in FAISS with dimension 768 per vector. RepLLaMA embeddings have a dimensionality of 4096 per vector and take 145GB space on disk. We assume they are fetched dynamically on-demand during retrieval. RetroMAE-2 [20] is not used because its checkpoint is not released.

For the MS MARCO Dev set, the result of a method is marked with tag † when statistically significant drop is observed compared to our method tagged with ▲ at 95% confidence level.

	MSMARCO Dev		**DL19**	**DL20**	**#Docs**	**Latency**	**Space**
	MRR	R@1k	NDCG	NDCG	**visited**	Total(ms)	GB
Full dense retrieval fused with SPLADE. Uncompressed flat index							
SPLADE (S)	0.396	0.980	0.732	0.721	8.8M	18.5	3.7
RepLLaMA (RL)	0.412	0.994	0.743	0.725	8.8M	–	145
S + RL	0.426	0.994	0.763	0.741	8.8M	–	149
RetroMAE (RM)	0.416	0.988	0.720	0.703	8.8M	1602.2	27.2
S + RM	0.425	0.988	0.740	0.731	8.8M	1620.7	30.9
CDFS vs. IVF for RepLLaMA (RL) under OPQ quantization							
RL	0.384†	0.990	0.719	0.707	8.8M	666.0	2.4
S + RL	0.417	0.991	0.760	0.747	8.8M	678.5	6.1
RL/IVF	0.365†	0.915†	0.710	0.694	3K	97.0	2.5
S + RL/IVF	0.393†	0.989	0.736	0.717	3K	115.5	6.2
RL/CDFS	0.387 †	0.980 †	0.730	0.716	3K	52.2	2.5
▲ S + RL/CDFS	0.419	0.988	0.760	0.753	3K	71.0	6.2
CDFS vs. IVF for RetroMAE (RM) under OPQ quantization							
RM	0.398†	0.984	0.701	0.702	8.8M	566.1	1.5
S + RM	0.416	0.988	0.737	0.732	8.8M	584.6	5.2
RM/IVF	0.308†	0.758†	0.558	0.581	8K	20.1	1.5
S + RM/IVF	0.349†	0.985	0.619	0.691	8K	38.6	5.2
RM/CDFS	0.395†	0.977†	0.702	0.681	4K	20.2	1.5
▲ S + RM/CDFS	0.415	0.986	0.740	0.730	4K	38.7	5.2

Table 1: In-memory cluster-based fusion with SPLADE

In-memory cluster-based partial dense retrieval fused with SPLADE. Table 1 compares CDFS and cluster-based partial dense retrieval baselines fused with SPLADE sparse retrieval under quantization. Here we assume the memory is constrained and data is compressed for in-memory dense retrieval computation. OPQ quantization in FAISS is configured with the number of codebooks as m =256 for RepLLaMA and m =128 for RetroMAE. IVF uses the same percentage of documents as CDFS for a fair comparison. Columns "Latency" and "Space" are the average single-query time and in-memory space cost of dense retrieval for MS MARCO Dev set. The top portion of this table lists the performance of stand-alone models and their fusion under full dense retrieval without compression.

The takeaway from Table 1 is that 1) CDFS delivers a relevance fairly close to full dense retrieval fused with sparse results under the same quantization setting while CDFS only searches about 3,000 out of 8.8M documents and is much faster. 2) The fusion with partial RepLLaMA dense retrieval under IVF and OPQ selecting about the same number of documents as CDFS can be fast in 115.5ms but the relevance is 7.6% worse in MRR@10 than CDFS which takes 71ms. The fusion with partial RetroMAE under IVF and OPQ is 15.9% worse in MRR@10 than CDFS while both spend around 38.6ms.

Data location	Rerank	LADR*	DiskANN	SPANN	CDFS
Embeddings	Disk	Disk	Disk/mem	Disk	Disk
Graph	–	Mem	Disk	–	–

	Relevance		**#Doc**	**Dense latency (ms)**			**Breakdown (ms)**	
	MRR	R@1k		MRT	P99	Worst	I/O	Comp.
SPLADE (S) + RepLLaMA								
Rerank	0.425	0.980	1000	298.7	377.6	402.5	294.1	4.6
LADR*$_{\text{fast}}$	0.420	0.979†	1107	127.6	377.3	445.2	115.8	11.8
LADR*$_{\text{default}}$	0.420	0.989	8522	894.8	2469.7	3430.1	837.9	56.5
▲ **S+CDFS**	0.425	0.988	2829	45.9	112.9	–	41.8	4.1
SPLADE (S) + RetroMAE								
Rerank	0.422	0.980	1000	14.1	108.3	423.17	13.3	0.9
LADR*$_{\text{fast}}$	0.422	0.982	1107	18.4	137.3	503.6	14.5	3.8
LADR*$_{\text{default}}$	0.425	0.988	8522	71.3	344.2	3876.7	45.4	25.9
DiskANN	0.402†	0.977	–	280.0	314.9	–	265.1	14.9
S+DiskANN	0.415†	0.985	–	280.0	314.9	–	265.1	14.9
SPANN	0.167†	0.753	–	145.2	135.1	–	–	10.1
S+SPANN	0.398†	0.981	–	145.2	135.1	–	–	10.1
▲ **S+CDFS**	0.424	0.987	4427	10.4	31.0	–	6.0	4.4

Table 2: Dense retrieval and fusion with on-disk embeddings. The time listed is only for dense retrieval.

CDFS vs. nearest-neighbor search and re-ranking baselines with in-memory or on-disk data. Table 2 assesses CDFS when MS MARCO passages data is hosted on SSD disk. The top portion of Table 2 specifies the data location of configured baselines. The reranking method simply fetches top-k embeddings from the disk for a fusion. LADR is designed for in-memory search [14], and we extend it as LADR* for on-disk search by accessing embeddings from the disk while assuming the memory can host a proximity graph. We test two configurations of LADR*, its default setting with 128 neighbors, exploration depth of 50, and 200 seed documents, as well as a fast configuration selected to have a similar CPU time as our method for RetroMAE. LADR*$_{\text{fast}}$ uses 128 neighbors, exploration depth 20, and 20 seed documents. DiskANN [11] assumes the graph and uncompressed embeddings are on disk while its memory hosts compressed embeddings for quick guidance. SPANN [3] searches disk data in a cluster-based manner following query-centroid distances. Both DiskANN and SPANN are designed to search from scratch, and we simply fuse their outcome with SPLADE results. We only present the results for RetroMAE because the officially released code for DiskANN and SPANN cannot build the RepLLaMA index in our machine due to memory limit. HNSW [26] is not included because its in-memory relevance is similar to DiskANN. The latency time reported is in milliseconds. The column "#Doc" is the number of documents fetched to perform dense retrieval.

Reranking does well in MRR@10 and has a slightly lower recall@1000. But it takes 298.7ms in RepLLaMA-based fusion, dominated by random 1,000 I/O operations to fetch embeddings. In comparison, CDFS issues 17 IO operations on average per query to fetch clusters. We find that each I/O operation has about 0.3ms startup cost as queuing and other software overhead in our tested PCIe SSD. Thus, more fine-grained operations in reranking yields more overhead, while CDFS, which fetches clusters of consecutive embeddings, utilizes block I/O that incurs less access overhead with fewer I/O operations issued. For the same reason, CDFS can accomplish similar and higher relevance compared to LADR* and DiskANN while having much less I/O cost. CDFS also outperforms SPANN as it only exploits query-centroid distances for selection.

Noted that our setting stores embeddings in similarity-based clusters on disk, thus random embedding-level access in reranking and LADR* still benefits from spatial locality in clustered data because top results tend to be similar. That explains why 99th percentile latency of re-ranking and LADR* is much slower when spatial locality is less available or not exploitable. When embedding vectors become longer from RetroMAE to RepLLaMA, clustered data locality becomes less beneficial, which explains why the ratio from 99th percentile latency over the mean latency drops from 7.7x to 1.27x. The column "Worst" lists the estimated worst-case latency when documents are not clustered on the disk.

	Stand-alone models			**SPLADE+RepLLaMA**			**SPLADE+RetroMAE**		
NDCG@10	SPLADE	RepLLaMA	RetroMAE	Full	Rerank	**CDFS**	Full	Rerank	**CDFS**
Avg.	0.500	0.551	0.482	0.561	0.545	0.554	0.520	0.483	0.517
- SPLADE	–	10.2%	-3.6%	12.2%	9.0%	10.8%	4.0%	-3.5%	3.5%

Table 3: Zero-shot retrieval performance on 13 BEIR datasets

BEIR. Table 3 lists average NDCG@10 on 13 BEIR datasets including DBPedia, FiQA, NQ, HotpotQA, NFCorpus, T-COVID, Touche, ArguAna, C-Fever, Fever, Quora, Scidocs, and SciFact. CDFS works effectively with RepLLaMA and RetroMAE after SPLADE.

Parameters	MRR@10	R@1K	#Clu.	MRR@10	R@1K	#Clu
(α, β), γ=0.06	**RepLlama+BM25**			**RetroMAE+BM25**		
(0.05,0)	0.410	0.967	20.9	0.408	0.955	20.9
(0,0.02)	0.410	0.972	28.2	0.412	0.967	27.2
(0,0.05)	0.413	0.976	53.9	0.414	0.971	44.0
(0.05,0.02)	0.412	0.973	34.5	0.414	0.969	30.2
(α, β), γ=0.03	**RepLlama+SPLADE**			**RetroMAE+SPLADE**		
(0.01,0)	0.420	0.983	4.2	0.408	0.980	4.2
(0.02,0)	0.423	0.985	7.1	0.415	0.981	7.2
(0,0.01)	0.425	0.988	16.2	0.423	0.987	14.8
(0.02,0.01)	0.425	0.988	16.8	0.424	0.987	15.5

Table 4: Varying α and β for direct and prob. guarantees

Parameter sensitivity. Table 4 illustrates the impact of using a few values of α and β parameters in controlling probabilistic and direct inclusion guarantees in CDFS. This table is for MS MARCO passage Dev set under four different sparse and uncompressed dense pairs with retrieval depth $k = 1000$. Column "#Clu" means the number of embedding clusters selected. The default γ value is 0.03 for SPLADE and 0.06 for BM25 sparse retrieval. The result shows that (α, β)=(0.02,0.01) is a well-balanced choice for SPLADE, and (0.05,0.02) for BM25, and the probabilistic guarantee adds diverse results to improve recall.

4 CONCLUDING REMARKS

The contribution of this paper is to demonstrate and evaluate how lightweight, cluster-based partial dense retrieval can achieve competitive relevance and short CPU latency without relying on a proximity graph. CDFS assumes that sparse retrieval runs first and judiciously selects a subset of document embeddings to conduct limited fusion. Our design prioritizes the dense search in document clusters based on a mixture of probabilistic and direct inclusion of top sparse results to quickly narrow the scope of searching relevant documents. When dense embeddings do not fit into memory, CDFS loads selected clusters from the disk storage in a block I/O manner, which is much faster than SPANN and graph navigation methods with finer-grained I/O such as DiskANN and an extension of LADR.

Acknowledgments. We thank Wentai Xie and anonymous referees for their help and valuable comments. This work was supported in part by NSF IIS-2225942 and ACCESS program. Any opinions, findings, or conclusions expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901.

[2] Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *NIPS* (2016).

[3] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.).

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2020 Deep Learning Track. *ArXiv* abs/2102.07662 (2020).

[5] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Term Weighting For First Stage Passage Retrieval. *SIGIR* (2020).

[6] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2022).

[7] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. *SIGIR* (2021).

[8] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. *NAACL* (2021).

[9] Luyu Gao, Zhuyun Dai, Zhen Fan, and J. Callan. 2021. Complementing Lexical Retrieval with Semantic Residual Embedding. *ECIR* abs/2004.13969 (2021).

[10] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In *Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23)*. ACM, New York, NY, USA, 3406–3416.

[11] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. 2019. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc.

[12] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[13] V. Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *EMNLP'2020* ArXiv abs/2010.08191 (2020).

[14] Hrishikesh Kulkarni, Sean MacAvaney, Nazli Goharian, and Ophir Frieder. 2023. Lexically-Accelerated Dense Retrieval. In *Proc. of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 152–162.

[15] Saar Kuzi, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Leveraging semantic and lexical matching to improve the recall of document retrieval systems: A hybrid approach. *arXiv preprint arXiv:2010.01195* (2020).

[16] Carlos Lassance and Stéphane Clinchant. 2022. An Efficiency Study for SPLADE Models. *SIGIR* (2022).

[17] Carlos Lassance, Simon Lupart, Hervé Déjean, Stéphane Clinchant, and Nicola Tonellotto. 2023. A Static Pruning Study on Sparse Neural Retrievers. In *Proc. of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 1771–1775.

[18] Hang Li, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and G. Zuccon. 2022. To Interpolate or not to Interpolate: PRF, Dense and Sparse Retrievers. *SIGIR* (2022).

[19] 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 (2021).

[20] Zheng Liu, Shitao Xiao, Yingxia Shao, and Zhao Cao. 2023. RetroMAE-2: Duplex Masked Auto-Encoder For Pre-Training Retrieval-Oriented Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 2635–2648.

[21] S. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489

[22] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. arXiv:2310.08319 [cs.IR]

[23] Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive Re-Ranking with a Corpus Graph. In *Proc. of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 1491–1500.

[24] Joel Mackenzie, Antonio Mallia, Alistair Moffat, and Matthias Petri. 2022. Accelerating Learned Sparse Indexes Via Term Impact Decomposition. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). ACL, Abu Dhabi, United Arab Emirates, 2830–2842.

[25] Joel Mackenzie, Matthias Petri, and Alistair Moffat. 2021. Anytime Ranking on Document-Ordered Indexes. *ACM Trans. Inf. Syst.* 40, 1, Article 13 (sep 2021), 32 pages.

[26] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (apr 2020), 824–836.

[27] Antonio Mallia, O. Khattab, Nicola Tonellotto, and Torsten Suel. 2021. Learning Passage Impacts for Inverted Indexes. *SIGIR* (2021).

[28] Antonio Mallia, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto. 2022. Faster learned sparse retrieval with guided traversal. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1901–1905.

[29] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant indexes and search for academia. *Proceedings of the Open-Source IR Replicability Challenge* (2019).

[30] Frederick Mosteller. 1946. On Some Useful "Inefficient" Statistics. *Annals of Mathematical Statistics* 17 (1946), 377–408.

[31] Yifan Qiao, Yingrui Yang, Shanxiu He, and Tao Yang. 2023. Representation Sparsification with Hybrid Thresholding for Fast SPLADE-based Document Retrieval. *ACM SIGIR'23* (2023).

[32] Yifan Qiao, Yingrui Yang, Haixin Lin, and Tao Yang. 2023. Optimizing Guided Traversal for Fast Learned Sparse Retrieval. In *Proceedings of the ACM Web Conference 2023*. 3375–3385.

[33] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3 (2009), 333–389.

[34] Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Xiaolong Huang, Binxing Jiao, Linjun Yang, and Daxin Jiang. 2023. LexMAE: Lexicon-Bottlenecked Pretraining for Large-Scale Retrieval. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=PfpEtB3-csK

[35] Michael Steinbach, George Karypis, and Vipin Kumar. 2000. A comparison of document clustering techniques. *TextMining Workshop at KDD2000* (2000).

[36] 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 *NeurIPS*.

[37] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[38] Wikipedia. 2023. Order Statistic. *https://en.wikipedia.org/wiki/Order_statistic* (2023).

[39] Shitao Xiao, Zheng Liu, Weihao Han, Jianjin Zhang, Defu Lian, Yeyun Gong, Qi Chen, Fan Yang, Hao Sun, Yingxia Shao, Denvy Deng, Qi Zhang, and Xing Xie. 2022. Distill-VQ: Learning Retrieval Oriented Vector Quantization By Distilling Knowledge from Dense Embeddings. *SIGIR* (2022).

[40] Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-training Retrieval-oriented Transformers via Masked Auto-Encoder. *EMNLP* (2022).

[41] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance. *CIKM* (2021).

[42] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval. In *Proc. of Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22)*. 1328–1336.