Hildoc: Leveraging Hilbert Curve Representation for Accurate and Efficient Document Retrieval

Muhammad Alqurishi
Elm Company
mualqurishi@elm.sa

Zhaozhi Qian
Elm Company
zqian@elm.sa

Faroq Altam
Elm Company
faltam@elm.sa

Riad Souissi
Elm Company
rsouissi@elm.sa

Abstract

Document retrieval is a critical challenge in information retrieval systems, where the goal is to efficiently retrieve relevant documents in response to a given query. Dense retrieval methods, which utilize vector embeddings to represent semantic information, require effective indexing to ensure fast and accurate retrieval. Existing methods, such as MEVI, have attempted to address this by using hierarchical K-Means for clustering, but they often face limitations in computational efficiency and retrieval accuracy. In this paper, we introduce the Hildoc Index, a novel document indexing approach that leverages the Hilbert Curve to map document embeddings onto a one-dimensional space. This innovative representation facilitates efficient clustering using a 1D quantile-based algorithm, ensuring uniform partition sizes and preserving the inherent structure of the data. As a result, Hildoc Index not only reduces training complexity but also enhances retrieval accuracy and speed during inference. Our method can be seamlessly integrated into both dense retrieval systems and hybrid ensemble systems. Through comprehensive experiments on standard benchmarks like MSMARCO Passage and Natural Questions, we demonstrate that the Hildoc Index significantly outperforms the current state-of-the-art MEVI in terms of both retrieval speed and recall. These results underscore the Hildoc Index as a solution for fast and accurate dense document retrieval.

1 Introduction

Document retrieval involves identifying relevant documents in response to a user's query (Li et al., 2024a,b). This process is a critical component of various applications, including search engines, AI recommendation systems, question-answering systems, and Agentic AI (Achiam et al., 2023; EduBrain, 2024; Kenton and Toutanova, 2019; Lewis et al., 2020).

A prevalent approach to document retrieval today is dense retrieval, which encodes documents and queries into dense vectors to capture semantic information (Zhang et al., 2021; Xiong et al., 2020). These vectors are then used with similarity metrics to rank candidate documents (Malkov and Yashunin, 2020; Douze et al., 2024; Karpukhin et al., 2020). However, searching through the entire document database for each query during inference is computationally prohibitive. This challenge has prompted researchers to develop various document indexing techniques to expedite the search process.

A document index partitions the document database into subsets and constructs dense representations for these partitions. This indexing method enhances document retrieval efficiency through a two-step approach. Initially, the top-C partitions most similar to the query are retrieved as a candidate set. Subsequently, documents within this candidate set are ranked based on their similarity to the query, with the top-k documents returned as results. This method significantly reduces computational demands by limiting similarity evaluations to the candidate set rather than the entire database.

The quality of the document index is pivotal to the accuracy and efficiency of the overall dense retrieval system. A well-constructed document index should meet three critical criteria. First, it must preserve the cluster structure within the document embeddings, ensuring that documents within the same cluster are likely to reside in the same partition (**Cluster Preservation**). This criterion guarantees that partitions respect the intrinsic structure of the embeddings, increasing the likelihood that relevant documents appear in the candidate set during the initial retrieval phase. Second, the document partitions should be of similar sizes (**Uniformity**). This uniformity prevents the inflation of the candidate set size, which could occur if an oversized partition is retrieved during the initial phase. Consequently, the retrieval system can maintain high computational efficiency across all queries at inference time. Lastly, the document index should be constructed efficiently during the training phase, exhibiting favorable average and worst-case time complexity (**Efficiency**). This criterion ensures that the indexing process does not become a bottleneck, allowing for scalable and practical deployment of the dense retrieval system.

However, it is challenging to design a retrieval system that achieve the three criteria. Prior work has shown a clear trade-off between cluster presentation, and uniformity and efficiency. For instance, HNSW and BATL leverage heuristic balanced tree index or graph index to achieve high uniformity and efficiency at the cost of cluster preservation (Malkov and Yashunin, 2020; Li et al., 2023). On the other hand, NCI and the state-of-the art retrieval method, MEVI, leverage hierarchical K-Means clustering algorithm to achieve high cluster preservation (Wang et al., 2022; Zhang et al., 2024). Yet, the K-Means algorithm often results in clusters of varying sizes, which undermines uniformity, and the repeated application of K-Means contributes to high training complexity, thereby compromising efficiency. There is yet to be a method that achieves all three criteria at the same time.

In this work, we introduce a novel principled document index for dense retrieval, Hildoc Index , by leveraging the Hilbert Curve representation of document embeddings. Our principal insight is that the Hilbert Curve's space-filling and measure-preserving properties enable a reduction in embedding dimensionality to 1D while preserving the inherent cluster structure of the data. Additionally, we employ a novel 1D quantile-based clustering algorithm that ensures high uniformity and enhances training efficiency. Collectively, these innovations allow the Hildoc index to facilitate more accurate and efficient document retrieval. Moreover, Hildoc can be integrated with other document retrieval methods in an ensemble configuration to further augment retrieval accuracy. We conduct comprehensive evaluations on two leading benchmarks, MSMARCO Passage and Natural Questions, demonstrating that Hildoc surpasses the state-of-the-art MEVI method in terms of both retrieval efficiency and accuracy.

Our contributions are summarized as follows:

1. We introduce the Hildoc Index (Hilbert Document Index), a novel document index for dense retrieval that utilizes Hilbert curve distances to facilitate cluster preservation, achieve high uniformity, and ensure efficient training.

2. Using Hildoc Index, we propose a dense retrieval system (Hildoc) and a hybrid retrieval system (Hildoc Ensemble).

3. Through extensive experimentation, we show that both Hildoc and Hildoc Ensemble achieve state-of-the-art (SOTA) performance in document retrieval efficiency and accuracy.

2 Dense retrieval and document index

2.1 A formal description of dense retrieval with document index

Let $\mathcal{D}$ be a collection of N documents, represented as $\mathcal{D} = \{x_i\}_{i=1}^{N}$. Given a query y, the objective is to identify the top-k related documents within $\mathcal{D}$. We operate in a dense retrieval setting, where both the documents in $\mathcal{D}$ and the query y are encoded into vector representations. Denote the document vector as $\mathbf{d}_i = f(x_i) \in \mathbb{R}^J$ for all $i \leq N$, where f is the encoding function (e.g., a twin-tower representation model), and $J \in \mathbb{N}^*$ is the dimensionality of the vector space. Similarly, denote the query vector as $\mathbf{q} = f(y) \in \mathbb{R}^J$.

In principle, one can compute the similarity score between the query and each document, denoted as $s_i = g(\mathbf{q}, \mathbf{d}_i)$, where g is the similarity function. A common choice for g is the inner product, $\mathbf{q} \cdot \mathbf{d}_i$, assuming the vectors are appropriately scaled. The documents with the top-k similarity scores are then returned. A major limitation of this approach is its computational cost, which becomes significant when N is large. A common solution is to employ a document index, as discussed below.

A document index partitions $\mathcal{D}$ into $M \in \mathbb{N}^*$ subsets, denoted as $\mathcal{D}_m$ for $m \in [M]$. Furthermore, let $\mathbf{r}_m \in \mathbb{R}^J$ represent the vector representation of the entire partition $\mathcal{D}_m$. The similarity between a *partition* and a query can be measured as $v_m = g(\mathbf{q}, \mathbf{r}_m)$.

With a document index, dense retrieval can be performed efficiently in a two-step process. First, the top-C partitions are retrieved based on the partition-query similarity scores v_m. Subsequently, the top-k documents are retrieved within the selected partitions $\mathcal{D}_m$ based on the document-query similarity scores s_i. This approach avoids the need to compute s_i for all documents in $\mathcal{D}$.

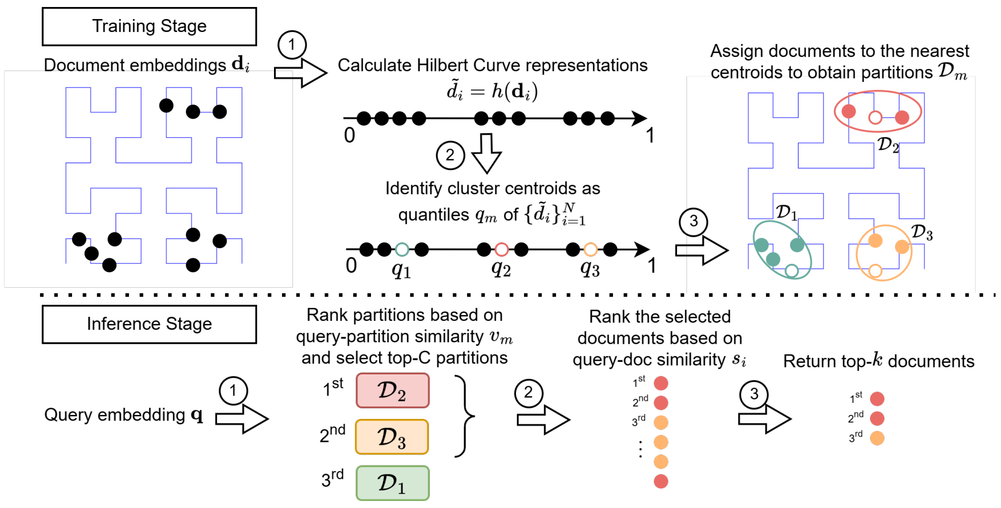
Training Stage
Document embeddings $\mathbf{d}_i$
1
Calculate Hilbert Curve representations
$\tilde{d}_i = h(\mathbf{d}_i)$
0
1
2
Identify cluster centroids as quantiles q_m of $\{\tilde{d}_i\}_{i=1}^N$
q_1
q_2
q_3
Assign documents to the nearest centroids to obtain partitions $\mathcal{D}_m$
$\mathcal{D}_1$
$\mathcal{D}_2$
$\mathcal{D}_3$
3
Inference Stage
Query embedding $\mathbf{q}$
Rank partitions based on query-partition similarity v_m and select top-C partitions
1st
2nd
3rd
Rank the selected documents based on query-doc similarity s_i
Return top-k documents

Figure 1: Illustration of training and inference with Hildoc Index. For illustrative purposes, we use two-dimensional embeddings ($J = 2$) of $N = 10$ documents to build Hildoc with $M = 3$ partitions. During inference, we retrieve top $C = 2$ partitions and $k = 3$ documents.

2.2 Criteria for an effective document index

As the document index plays a crucial role in dense retrieval, we consider the following key criteria when developing document indices to ensure accurate and efficient retrieval.

Cluster Preservation. Document vectors often exhibit a clustering structure characterized by two properties: Homogeneity and Separation. Assume there exist K clusters $\mathcal{C}_1, \dots, \mathcal{C}_K$ with respective centroids $\mathbf{e}_1, \dots, \mathbf{e}_K$. Homogeneity implies that documents within a cluster are close to its centroid:

$$\text{Homogeneity: } ||\mathbf{d}_i - \mathbf{e}_k|| \leq \delta_h, \ \forall k \in [K], \ i \in \mathcal{C}_k, \quad (1)$$

for a constant δ_h. A small δ_h (indicating high homogeneity) suggests that the cluster centroid $\mathbf{e}_k$ accurately represents all documents within the cluster. The second property, Separation, indicates that clusters are separated by at least a constant δ_s (larger δ_s indicates better-separated clusters):

$$\text{Separation: } ||\mathbf{d}_i - \mathbf{d}_j|| \geq \delta_s, \quad (2)$$

for all $i \in \mathcal{C}_{k_1}, \ j \in \mathcal{C}_{k_2}, \ k_1 \neq k_2$.

It is desirable for the document index to preserve this clustering structure by setting the partition representation as cluster centroids $\mathbf{r}_m = \mathbf{e}_k$ and assigning documents according to the clusters $\mathcal{D}_m = \mathcal{C}_k$. During the initial retrieval phase, Homogeneity and Separation ensure that only documents highly similar to $\mathbf{r}_m$ are retrieved, thereby enhancing accuracy.

However, directly clustering document vectors $\mathbf{d}_i$, $i \in [N]$, is computationally expensive, particularly when N is large, thus violating the Efficiency criterion discussed below. The Hildoc Index is designed to preserve clustering structure while remaining computationally efficient.

Uniformity. Uniformity ensures that partitions do not become overly large, which would make the index less efficient. It is defined by bounding the maximum partition size by a constant δ_u:

$$\max |\mathcal{D}_m| \leq \delta_u. \quad (3)$$

This property is critical for inference time efficiency, as it prevents the retrieval of disproportionately large partitions, which would take a long time to process.

Efficiency. Efficiency pertains to the computational demands during the training phase (when the index is constructed). Given the large size of the document database $\mathcal{D}$, it is essential to employ algorithms that scale well with respect to N.

3 Hildoc Index

The Hildoc Index offers a novel and principled approach to constructing a document index that meets the aforementioned criteria, thereby facilitating accurate and efficient retrieval. To construct the Hildoc Index, we map the vector representations $\mathbf{d}$ into a one-dimensional representation $\tilde{d}$ using the Hilbert Curve. Subsequently, we apply quantile-based clustering to these representations to derive the document index. The pseudo-code of Hildoc Index is shown in Appendix C.

3.1 Computing the Hilbert Curve representation of vectors

The Hilbert Curve is a continuous mapping from a J-dimensional hypercube to a one-dimensional interval, expressed as $h : [0, 1]^J \rightarrow [0, 1]$. To construct the Hildoc Index, we compute the Hilbert Curve representation of a document vector as $\tilde{d}_i = h(\mathbf{d}_i)$. It is crucial to note that the one-dimensional representation $\tilde{d}_i$ preserves the cluster structure inherent in the original vector $\mathbf{d}_i$ (Moon et al., 2001). Consequently, clustering can be performed on $\tilde{d}_i \in [0, 1]$ instead of $\mathbf{d}_i \in \mathbb{R}^J$, as we show below.

Proposition 1. *If the document vectors $\mathbf{d}_i$ exhibit Homogeneity and Separation with constants δ_h and δ_s, respectively, as defined in Equations 1 and 2, then their Hilbert Curve representations $\tilde{d}_i$ also exhibit Homogeneity and Separation, characterized by the following constants:*

$$\begin{aligned} \textit{Homogeneity: } & |\tilde{d}_i - \tilde{e}_k| \leq \omega_1 \cdot \delta_h^{1/J}, \\ & \forall k \in [K],\ i \in \mathcal{C}_k, \\ \textit{Separation: } & |\tilde{d}_i - \tilde{d}_j| \geq \omega_2 \cdot \delta_s^J, \forall i \in \mathcal{C}_{k_1}, \\ & j \in \mathcal{C}_{k_2},\ k_1, k_2 \in [K],\ k_1 \neq k_2 \end{aligned}$$

where $\omega_1, \omega_2 \in \mathbb{R}^+$ are positive constants and $\tilde{e}_k = h(\mathbf{e}_k)$.

Proposition 1 is derived from the bi-Lipschitz properties and Hölder continuity of Hilbert Curves. The proof is provided in Appendix B.

In practice, Hilbert Curves are computed iteratively (Appendix E). We denote the Hilbert Curve at iteration t as h^t, where $t < T$, the maximum number of iterations. A well-known property is that the length of the Hilbert Curve is $2^t - 2^{-t}$, which increases exponentially with iteration t. This implies that the Hilbert Curve can achieve high numerical precision after only a few iterations (Fisher, 1986).

3.2 Building Hildoc Index via quantile-based one-dimensional clustering

Having represented documents along the Hilbert Curve, the next step is to cluster these documents to form the Hildoc Index. We employ a clustering method based on quantiles, which are well-defined and straightforward to compute for one-dimensional data. The interval $[0, 1]$, which contains the Hilbert Curve representations $\{\tilde{d}_i\}_{i=1}^N$, is divided into segments by the cutoff points $[q_1, \ldots, q_M]$. These cutoff points, q_m, are selected such that there are exactly N/M representations $\tilde{d}_i$ within each segment $[q_m, q_{m+1})$. In practice, the q_m values are derived from the empirical quantiles of $\tilde{d}_i$, ensuring that each q_m corresponds to a document in the corpus, i.e., for each $m \in \{1, \ldots, M\}$, there exists an $i \in \{1, \ldots, N\}$ such that $\tilde{d}_i = q_m$.

The documents located at these cutoff points serve as the cluster centroids. It is important to note that for any given m, there are N/M documents in both the left segment $[q_{m-1}, q_m)$ and the right segment $[q_m, q_{m+1})$. Thus, q_m effectively acts as the "center" of the combined segment $[q_{m-1}, q_{m+1})$. The document vector at the cutoff point is used as the representation for the partition, i.e., for each $m \in \{1, \ldots, M\}$, $\mathbf{r}_m = \mathbf{d}_i$ for the document i such that $\tilde{d}_i = q_m$.

To complete the construction of the Hildoc Index, documents must be assigned to partitions. For each document i, we identify the segment it belongs to, specifically finding m such that $\tilde{d}_i \in [q_m, q_{m+1})$. We consider partitions m and $m + 1$ as potential assignments. The similarity between the document and the two partitions is compared using a similarity function $g(\mathbf{d}_i, \mathbf{r}_m)$ and $g(\mathbf{d}_i, \mathbf{r}_{m+1})$. The document is then assigned to the partition with the greater similarity.

The quantile-based 1D clustering approach facilitates Uniformity, as detailed in Proposition 2.

Proposition 2. *The Hildoc Index achieves worst-case uniformity as follows:*

$$\textit{Uniformity of Hildoc Index: } \max |\mathcal{D}_m| \leq \frac{2N}{M}.$$

Proposition 2 is derived from the fact that partition m may only contain documents from the segment $[q_{m-1}, q_{m+1})$, which encompasses a total of $2N/M$ documents. It is noteworthy that if documents are partitioned evenly, we have $\max |\mathcal{D}_m| = N/M$, which represents optimal uniformity. In comparison, the worst-case partition size for the Hildoc Index is only twice as large as the optimal scenario. Many clustering algorithms, such as K-Means and Hierarchical Clustering, exhibit worst-case uniformity of $\max |\mathcal{D}_m| \leq N - M + 1$, resulting in one large cluster and $M - 1$ clusters containing a single element each. By employing quantile-based one-dimensional clustering, the Hildoc Index avoids these adverse scenarios that could degrade retrieval speed.

Method	Reference	Cluster Preservation Homogenity	Cluster Preservation Separation	Uniformity	Efficiency
HNSW	(Malkov and Yashunin, 2020)	✗	✗	✓	✓
BATL	(Li et al., 2023)	✓	✗	✓	✗
NCI	(Wang et al., 2022)	✓	✓	✗	✗
MEVI	(Zhang et al., 2024)	✓	✓	✗	✗
Hildoc	This work	✓	✓	✓	✓

Table 1: Comparison with prior document retrieval methods. Hildoc is the first approach that simultaneously achieves cluster preservation, uniformity, and efficiency.

3.3 Efficiency of the Hildoc Index

The most computationally expensive step in constructing the Hildoc Index is determining the cluster centroids. One potential approach is to sort the values $\tilde{d}_i$ and select the cutoff points directly from the sorted sequence. With an appropriate algorithm, sorting can be achieved in $O(N \log N)$ time complexity for both average and worst-case scenarios (Appendix D). In contrast, MEVI proposed by (Zhang et al., 2024) employs hierarchical K-Means, which incurs an average-case complexity of $O(NMJLH)$, where L is the number of iterations required for K-Means convergence, and H represents the number of hierarchy levels. This complexity becomes significant in practice when M is approximately one million and L typically reaches a few thousand iterations (Zhang et al., 2024). Moreover, K-Means exhibits a substantially higher worst-case complexity of $O(N^{M+2/J})$.

3.4 Applying Hildoc Index in dense retrieval

The Hildoc Index can serve as a modular component in a two-phase dense retrieval system, as detailed in Section 2.1. We refer to the resulting system as the Hildoc. Additionally, the Hildoc Index can be integrated with another document retrieval method to form an ensemble (Zhang et al., 2024). Specifically, the other method is required to return it's own candidate documents and their relevance rankings. A common ensemble strategy involves combining the documents retrieved by both methods into a unified candidate set, which is re-ranked based on weighted similarity:

$$\bar{s}_i = s_i + \alpha \cdot \frac{1}{\beta \cdot r_i + 1}, \qquad (4)$$

where s_i denotes the similarity score produced by Hildoc, r_i represents the document ranking assigned by the other method, and $\alpha, \beta \in \mathbb{R}^+$ are hyperparameters. The documents with the top-k similarity scores $\bar{s}_i$ in this combined candidate set are returned as the final result.

4 Related Work

Document retrieval is a critical task that involves identifying relevant documents in response to a user's query (Li et al., 2024a,b). Existing approaches are categorized into three main types: sparse retrieval (Bevilacqua et al., 2022; Dai and Callan, 2019), dense retrieval (Karpukhin et al., 2020; Xiong et al., 2020), and generative retrieval (Metzler et al., 2021; Tay et al., 2022).

Traditional sparse retrieval methods, such as TF-IDF, rely on partial string matching and accelerate search processes using inverted indices like BM25 (Robertson and Zaragoza, 2009), SPLADE (Formal et al., 2021), and UniCOIL (Lin and Ma, 2021). While these methods are efficient, they often fall short in capturing semantic nuances, leading to incorrect responses when synonymous terms are used.

Dense retrieval methods address this limitation by encoding documents and queries into dense vectors that encapsulate semantic information, using similarity metrics to rank candidate documents. The primary challenge in document retrieval lies in constructing a document index that ensures rapid query responses while maintaining high accuracy. Table 1 illustrate the features of prior works through the lens of the three desiderata. Prior works have explored various indexing structures, including balanced tree indices such as BATL (Li et al., 2023), neighbor graph indices such as HNSW (Malkov and Yashunin, 2020), and hybrid approaches (Chen et al., 2021). However, these approaches do not explicitly preserve the clustering structure, which could lead to lower retrieval recall. Recently, (Zhang et al., 2024) introduced the MEVI method, which innovatively employs residual quantization (RQ) codebooks to facilitate hierarchical K-Means clustering. However, MEVI's cluster preservation comes at a cost for Efficiency and Uniformity as shown in Table 1.

Unlike dense retrieval, generation-based re-

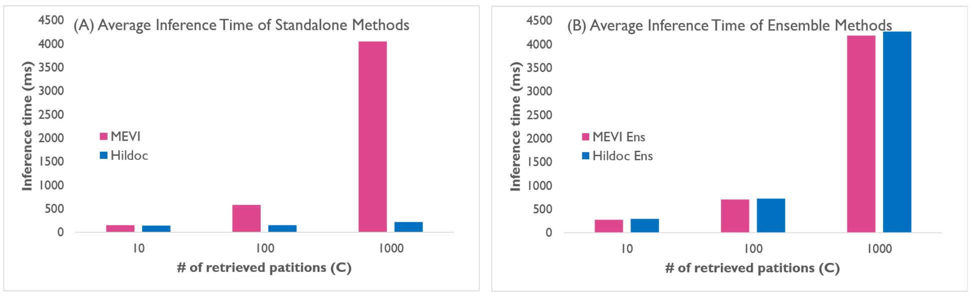

Figure 2: Average inference time on Natural Questions with varying retrieved partitions C.

Method	Natural Questions				MSMARCO			
	MRR@10	MRR@50	R@1	R@50	MRR@10	MRR@50	R@1	R@50
MEVI (C-10)	51.48	51.79	45.40	69.64	33.66	34.19	22.93	63.79
MEVI (C-100)	60.45	60.75	53.41	80.03	38.11	38.96	25.02	81.15
MEVI (C-1000)	64.53	64.85	56.79	85.68	39.26	40.20	25.77	85.52
MEVI Ens (C-10)	63.55	63.87	55.79	85.18	37.00	37.97	24.19	82.53
MEVI Ens (C-100)	63.42	63.75	55.49	86.40	38.39	39.34	25.08	84.52
MEVI Ens (C-1000)	64.64	64.97	56.62	87.09	39.00	39.94	25.34	85.76
Hildoc (C-10)	60.19	60.51	52.19	81.55	33.97	34.75	22.20	72.44
Hildoc (C-100)	64.05	64.37	56.23	85.73	37.49	38.40	24.57	81.76
Hildoc (C-1000)	65.40	65.72	57.31	87.20	38.87	39.81	25.45	85.05
Hildoc Ens (C-10)	63.81	64.09	56.23	84.40	37.80	38.69	24.96	81.06
Hildoc Ens (C-100)	65.68	65.99	57.95	87.34	39.30	40.26	25.90	86.15
Hildoc Ens (C-1000)	**66.32**	**66.61**	**58.34**	**88.14**	**39.62**	**40.60**	**26.00**	**87.01**

Table 2: Retrieval accuracy of different methods on Natural Questions and MSMARCO. The C value denotes the number of retrieved partitions.

trieval methods employ deep auto-regressive models to directly generate document identifiers from query inputs(Bevilacqua et al., 2022; Tay et al., 2022; Zhang et al., 2024; Wang et al., 2022; Zhou et al., 2023). However, these methods often require auxiliary processes, such as clustering or indexing, to create training datasets for the auto-regressive models. For instance, NCI (Wang et al., 2022) utilizes hierarchical K-means clustering, while SEAL (Bevilacqua et al., 2022) employs the FM-Index on document n-grams (Ferragina and Manzini, 2000). Consequently, advancements in document indexing remain a crucial, complementary effort that can significantly enhance the training phase of generation-based retrieval methods.

5 Experiments

5.1 Experimental Settings

To maintain consistency and facilitate comparison with related studies, we employ the standard benchmark settings commonly used in document retrieval (Zhang et al., 2024).

Benchmark Datasets. To evaluate the efficiency and accuracy of the proposed method, we benchmarked Hildoc on two large-scale passage retrieval datasets: Natural Questions (NQ) (Kwiatkowski et al., 2019) and MSMARCO Passage (Nguyen, 2016). The NQ dataset is an open-domain question answering dataset containing 21,015,324 passages and 3,610 test queries, collected from Google search logs. The MSMARCO dataset is a smaller dataset containing 8,841,823 passages and 6,980 queries, developed by Microsoft.

Evaluation Metrics. We utilized two widely recognized metrics, R@K and MRR@K, to assess the relevance of retrieved documents to the input queries. R@K (Recall at K) represents the proportion of relevant documents among the top-K retrieved documents. MRR@K (Mean Reciprocal Rank at K) evaluates the method's ability in retrieving relevant documents with higher ranks (Zhang et al., 2024).

Baselines. We compare Hildoc with the state-of-the-art (SOTA) baseline MEVI (Zhang et al., 2024). Our focus is on top-performing models, as weaker baselines such as BM25, SPLADE, HNSW, and

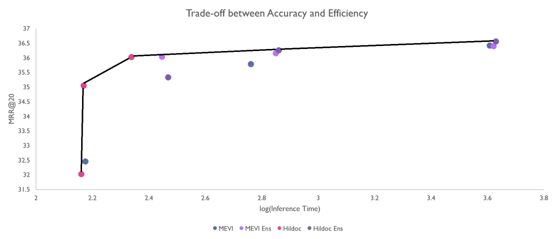

Figure 3: Trade-off between accuracy (MRR@20) and efficiency (inference time) on Natural Questions by selecting different number of retrieved partitions C.

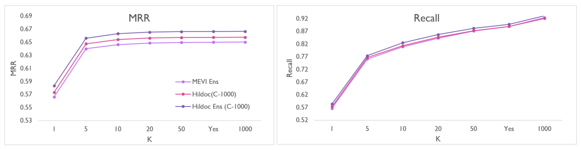

Figure 4: Performance curve for MRR and Recall across different k values on the Natural Questions.

NCI have been shown to underperform MEVI in the standard benchmark setting we have adopted (Zhang et al., 2024). For ensemble methods, we consider the ensemble of MEVI and HNSW as a baseline, originally proposed and validated in (Zhang et al., 2024).

Embedding Models. We employed two distinct embedding methods, T5-ANCE (OpenMatch, 2022) and AR2 (Zhang et al., 2021), to generate embedding vectors. Both methods have demonstrated effectiveness in prior studies. These methods utilize different backbone models, providing a means to test the applicability of our method to various twin-tower representation models.

Computational resources. The experiments were conducted on a single A6000 GPU.

5.2 Results on retrieval efficiency

Computational efficiency during inference time is a crucial aspect in practical document retrieval applications. Since both MEVI and Hildoc partitions the documents by clustering, we can perform an apple-to-apple comparison of their retrieval efficiency by varying the number of partitions retrieved, C. Figure 2 shows the speed of retrieval on Natural Questions dataset in millisecond for $C = 10, 100, 1000$. Empirically the inference time of MEVI increased dramatically with the increase in the number of retrieved clusters, reaching an unpractical 4000 ms when $C = 1000$. In contrast, Hildoc's efficiency is consistently high across various C values, and always less then 200 ms.

The results above suggest that Hildoc employs a better clustering technique to ensure high Uniformity (Proposition 2). In contrast, MEVI's hierarchical K-Means clustering appears to less uniform - and when a larger number of partitions are retrieved, it is more likely to hit a very large partition, thereby incurring high computational cost. The MEVI and Hildoc ensembles have similar inference time since the time is determined by the slowest method in an ensemble.

5.3 Results on retrieval accuracy

Table 2 presents the retrieval accuracy, evaluated using the Mean Reciprocal Rank (MRR) and Recall metrics (More experimental results are presented in Appendix F). Recall from the previous section that the standalone Hildoc with 1000 partitions is even faster than MEVI with 100 partitions. Hence a fair and practically relevant comparison should compare Hildoc C-1000 with MEVI C-100, and not with MEVI C-1000. Indeed, the standalone Hildoc C-100 outperforms both the MEVI and MEVI Ensemble models across all C levels on the Natural Questions dataset. Its performance further improves when retrieving 1000 partitions. On the MSMARCO Passage dataset, Hildoc exceeds the MEVI Ensemble's performance when C=1000, and it demonstrates performance comparable to the MEVI Ensemble when retrieving only 100 partitions.

Turning to the Hildoc Ensemble, it achieves the highest overall performance among all methods compared when retrieving 1000 partitions. Notably, the Hildoc Ensemble matches or surpasses the performance of the MEVI Ensemble with the same C value. Additionally, the Hildoc Ensemble with C=100 achieves performance roughly equivalent to that of the standalone Hildoc retrieving 1000 partitions. These results indicate that both Hildoc and Hildoc Ensemble can accurately identify relevant documents while retrieving a modest number of partitions, compared to state-of-the-art baselines.

Consistent performance improvement across various k values. Table 2 demonstrates Hildoc's consistent performance pattern for MRR and Recall at different k values, representing the number of top-k retrieved documents. We present results for a broad range of k values (from 1 to 1000) to confirm the consistency of Hildoc's advantage. Figure 4 illustrates the performance curve for MRR and Recall across different k values on the Natural

Hilbert curve order	MRR@10	MRR@20	R@50	R@100	R@1000
$T = 4$	35.24	35.91	81.24	86.50	94.25
$T = 7$	35.16	35.82	81.20	86.38	94.15
$T = 15$	35.36	36.03	81.64	86.76	94.73
Largest Gap (%)	0.57%	0.57%	0.54%	0.43%	0.62%

Table 3: Impact of Hilbert Curve order T on Hildoc's performance on MSMARCO

Embedding Model	MRR@10	MRR@20	R@50	R@100	R@1000
T5-ANCE	35.36	36.03	81.64	86.76	94.73
AR2	38.78	39.43	84.93	89.63	95.54

Table 4: Impact of Embedding model (T5-ANCE and AR2) on Hildoc's performance on MSMARCO

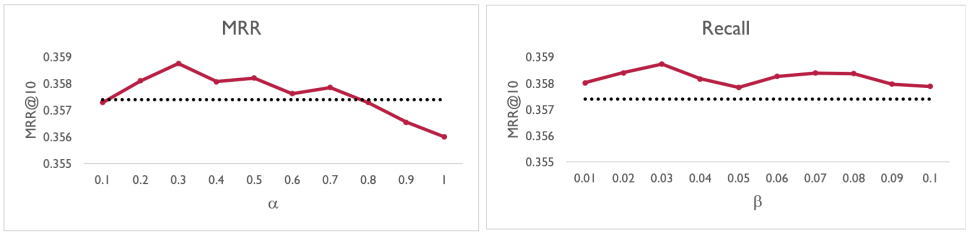

Figure 5: Impact of Hyperparameters α, β on Hildoc Ensemble's performance on MSMARCO.

Questions dataset. As depicted in the figure, the Hildoc Ensemble consistently outperforms other methods, followed by standalone Hildoc.

5.4 Trade-off between accuracy and efficiency

Figure 3 depicts the trade-off between accuracy (MRR@20) and efficiency (inference time) for various methods. We computed results for Hildoc, MEVI, and their ensembles by varying the number of retrieved partitions C. The Pareto front of accuracy and efficiency is highlighted with a black line. Methods on the Pareto front are considered "optimal" since no other method simultaneously surpasses them in both accuracy and efficiency (Sawaragi et al., 1985). Notably, the Pareto front is formed by Hildoc or Hildoc Ensemble, with standalone Hildoc excelling in the low inference time regime and Hildoc Ensemble in the high inference time regime. MEVI and its ensemble do not lie on the Pareto front, indicating that Hildoc can outperform them in both dimensions.

5.5 Hyperparameters and sensitivity

Impact of Hilbert Curve order T. To assess the effect of the Hilbert curve order T on model performance, we evaluated the Hildoc algorithm with $T = 4, 7$, and 15 on the MSMARCO passage dataset. Table 3 shows that the highest performance is achieved with $T = 15$. However, the performance variability is minimal (the percentage difference between the best and worst performance is within 0.62%). It is important to note that the Hilbert curve iteration T only affects training speed. Therefore, we recommend using a larger T during training when feasible.

Impact of Embedding Model. Table 4 presents the performance of Hildoc using two different embedding models, T5-ANCE and AR2. We observe that Hildoc with AR2 consistently outperforms the version with T5-ANCE, highlighting the critical role of the embedding model in dense retrieval. This pattern is also observed in MEVI and its ensembles, suggesting that AR2 is generally a superior embedding model compared to T5-ANCE and should be preferred in practical applications.

The Ensemble Hyperparameters α, β. We evaluated the hyperparameters on a grid with $\alpha \in [0.1, 1]$ and $\beta \in [0.01, 0.1]$ to assess model performance on MSMARCO. We found that $\alpha = 0.3$ and $\beta = 0.03$ achieved the best performance. Figure 5 displays the MRR@10 score when varying α and β while holding the other constant. Additionally, the best-performing hyperparameters are approximately 2% better than the worst-performing ones, suggesting that tuning ensemble hyperparameters may unlock some performance improvement, though not substantial.

6 Limitations

In this study, we propose Hildoc, a novel dense embedding retrieval method that improves over existing methods on retrieval accuracy and efficiency. One limitation of our theoretical analysis (Proposition 1) is that the bounds may become loose when the embedding dimensionality J is large, which represents the "curse of dimensionality". However, our experiments confirms Hildoc's practical performance improvement with embeddings generated by two widely used embedding models (T5-ANCE and AR2).

References

J. Achiam and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

M. Bevilacqua and 1 others. 2022. Autoregressive search engines: Generating substrings as document identifiers. *Advances in Neural Information Processing Systems*, 35:31668–31683.

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. *Advances in Neural Information Processing Systems*, 34:5199–5212.

Z. Dai and J. Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687*.

M. Douze and 1 others. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

EduBrain. 2024. Edubrain. `https://edubrain.org/`.

P. Ferragina and G. Manzini. 2000. Opportunistic data structures with applications. In *Proceedings 41st annual symposium on foundations of computer science*. IEEE.

AJ Fisher. 1986. A new algorithm for generating hilbert curves. *Software: Practice and Experience*, 16(1):5–12.

T. Formal, B. Piwowarski, and S. Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

H.V. Jagadish. 1990. Linear clustering of objects with multiple attributes. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*.

V. Karpukhin and 1 others. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

J.D.M.-W.C. Kenton and L.K. Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, Minneapolis, Minnesota.

T. Kwiatkowski and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

W. Li and 1 others. 2023. Learning balanced tree indexes for large-scale vector retrieval. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

X. Li and 1 others. 2024a. From matching to generation: A survey on generative information retrieval. *arXiv preprint arXiv:2404.14851*.

Y. Li and 1 others. 2024b. A survey of generative search and recommendation in the era of large language models. *arXiv preprint arXiv:2404.16924*.

J. Lin and X. Ma. 2021. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. *arXiv preprint arXiv:2106.14807*.

Y.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):824–836.

D. Metzler and 1 others. 2021. Rethinking search: making domain experts out of dilettantes. In *ACM SIGIR Forum*. ACM New York, NY, USA.

B. Moon and 1 others. 2001. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on knowledge and data engineering*, 13(1):124–141.

T. Nguyen. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

OpenMatch. 2022. T5-ance. `https://huggingface.co/OpenMatch/t5-ance`.

Owen O'malley. 2008. Terabyte sort on apache hadoop. *Yahoo, available online at: http://sortbenchmark.org/Yahoo-Hadoop. pdf,(May)*, pages 1–3.

S. Robertson and H. Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Yoshikazu Sawaragi, HIROTAKA NAKAYAMA, and TETSUZO TANINO. 1985. *Theory of multiobjective optimization*, volume 176. Elsevier.

John Skilling. 2004. Programming the hilbert curve. In *AIP Conference Proceedings*, volume 707, pages 381–387. American Institute of Physics.

Y. Tay and 1 others. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843.

Y. Wang and 1 others. 2022. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35:25600–25614.

L. Xiong and 1 others. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

H. Zhang and 1 others. 2021. Adversarial retriever-ranker for dense text retrieval. *arXiv preprint arXiv:2110.03611*.

H. Zhang and 1 others. 2024. Model-enhanced vector index. *Advances in Neural Information Processing Systems*, 36.

Y.-J. Zhou and 1 others. 2023. Dynamicretriever: a pre-trained model-based ir system without an explicit index. *Machine Intelligence Research*, 20(2):276–288.

A Notations

Table 5.

B Proofs

Proposition 1. *If the document vectors $\mathbf{d}_i$ exhibit Homogeneity and Separation with constants δ_h and δ_s, respectively, as defined in Equations 1 and 2, then their Hilbert Curve representations $\tilde{d}_i$ also exhibit Homogeneity and Separation, characterized by the following constants:*

$$\textit{Homogeneity: } |\tilde{d}_i - \tilde{e}_k| \leq \omega_1 \cdot \delta_h^{1/J}, \quad \forall k \in [K],\ i \in \mathcal{C}_k,$$
$$\textit{Separation: } |\tilde{d}_i - \tilde{d}_j| \geq \omega_2 \cdot \delta_s^{J},\ \forall i \in \mathcal{C}_{k_1},\ j \in \mathcal{C}_{k_2},\ k_1, k_2 \in [K],\ k_1 \neq k_2$$

where $\omega_1, \omega_2 \in \mathbb{R}^+$ are positive constants and $\tilde{e}_k = h(\mathbf{e}_k)$.

Proof of Homogeneity. Recall that the definition of Homogenetiy ensures that (Equation 1 reproduced below)

$$||\mathbf{d}_i - \mathbf{e}_k|| \leq \delta_h,\ \forall k \in [K],\ i \in \mathcal{C}_k. \quad \text{(B.1)}$$

To show Homogeneity on Hilbert Curve, we need to find an upper bound of $|\tilde{d}_i - \tilde{e}_k|$. We take advantage of Hilbert Curves' Hölder continuity property (Jagadish, 1990), given that $\tilde{d}_i$ and $\tilde{e}_k$ are the Hilbert Curve representations of $\mathbf{d}_i$ and $\mathbf{e}_k$ respectively. For all $k \in [K]$, $i \in \mathcal{C}_k$, the following equation holds:

$$\text{Hölder continuity: } \exists \omega_1 > 0,\ s.t.\ |\tilde{d}_i - \tilde{e}_k| \leq \omega_1 \cdot ||\mathbf{d}_i - \mathbf{e}_k||^{1/J} \quad \text{(B.2)}$$

Combining Equation B.1 and B.2, we obtain the Homogeneity on Hilbert Curve.

Proof of Separation. Recall that the definition of Separation ensures that (Equation 2 reproduced below)

$$||\mathbf{d}_i - \mathbf{d}_j|| \geq \delta_s,\ \forall i \in \mathcal{C}_{k_1},\ j \in \mathcal{C}_{k_2},\ k_1 \neq k_2, \quad \text{(B.3)}$$

To show Separation on Hilbert Curve, we need to find an lower bound of $|\tilde{d}_i - \tilde{d}_j|$. We take advantage of Hilbert Curves' Forward locality property (Jagadish, 1990), given that $\tilde{d}_i$ and $\tilde{d}_j$ are the Hilbert Curve representations of $\mathbf{d}_i$ and $\mathbf{d}_j$ respectively. For all $i \in \mathcal{C}_{k_1}$, $j \in \mathcal{C}_{k_2}$, $k_1 \neq k_2$, the following equation holds:

$$\exists C > 0\, s.t.\ ||\mathbf{d}_i - \mathbf{d}_i|| \leq C \cdot |\tilde{d}_i - \tilde{d}_j|^{1/J} \quad \text{(B.4)}$$

Re-arranging the terms and raising both sides to the power of J, we get

$$|\tilde{d}_i - \tilde{d}_j| \geq \frac{1}{C^J} \cdot ||\mathbf{d}_i - \mathbf{d}_i||^J \quad \text{(B.5)}$$

Combining Equation B.5 and B.3 we obtain the Separation property on Hilbert Curve, where the constant $\omega = \frac{1}{C^J}$

We refer the readers to Jagadish (1990) for a more in-depth discussion about the cluster preservation properties of Hilbert Curve representations.

C Pseudo-code of Hildoc Index

Algorithm 1 Training Hildoc Index

Input: document database $\mathcal{D}$, number of partitions M.

Output: partitions $\mathcal{D}_1 \dots \mathcal{D}_M$ and partition vectors $\mathbf{r}_1 \dots \mathbf{r}_M$

```
1:  for $i = 1$ to $N$ do
2:      $\mathbf{d}_i \leftarrow f(x_i)$
3:      $\tilde{d}_i \leftarrow h^t(\mathbf{d}_i)$ {Compute Hilbert Curve Representations for each document}
4:  end for
5:  $\{l_i\}_{i=1}^N \leftarrow argsort(\{\tilde{d}_i\}_{i=1}^N)$
6:  $m = 1$
7:  for $i = 1$ to $N$ do
8:      if $i = \frac{mN}{M}$ then
9:          $q_m \leftarrow \tilde{d}_j$ where $j = l_i$ {The quantiles of $\{\tilde{d}_i\}_{i=1}^N$}
10:         $\mathbf{r}_m \leftarrow \mathbf{d}_j$ where $j = l_i$ {Compute the cluster centroids}
11:         $m \leftarrow m + 1$
12:     end if
13: end for
14: for $i = 1$ to $N$ do
15:     $\mathbf{d} \leftarrow \mathbf{d}_{l_i}$ {Retrieve the embedding of the $i$-th smallest document on the Hilbert Curve}
16:     $c_1 \leftarrow floor(\frac{iM}{N}), c_2 \leftarrow ceil(\frac{iM}{N})$
17:     $\mathbf{e}_1 \leftarrow \mathbf{r}_{c_1}$, $\mathbf{e}_2 \leftarrow \mathbf{r}_{c_2}$ {Obtain the two candidate cluster centroids}
18:     $v_1 \leftarrow g(\mathbf{d}, \mathbf{e}_1)$, $v_2 \leftarrow g(\mathbf{d}, \mathbf{e}_2)$
19:     if $v_1 < v_2$ then
20:         $\mathcal{D}_{c_1} \leftarrow \mathcal{D}_{c_1} \cup \{x_{l_i}\}$
21:     else
22:         $\mathcal{D}_{c_2} \leftarrow \mathcal{D}_{c_2} \cup \{x_{l_i}\}$
23:     end if
24: end for
25: Return $\mathcal{D}_1 \dots \mathcal{D}_M$ and $\mathbf{r}_1 \dots \mathbf{r}_M$
```

Notation	Value Range	Explanation
x_i	Document	Individual document in the collection
$\mathcal{D}$	$\{x_i\}_{i=1}^{N}$	Collection of N documents
y	Query	The query for which related documents are sought
$\mathbf{d}_i$	$\mathbb{R}^J$	Vector representation of document x_i
$\tilde{d}_i$	$[0, 1]$	Hilbert Curve representation of document x_i
$\mathbf{q}$	$\mathbb{R}^J$	Vector representation of the query y
f	Function	Encoding function for documents and queries
J	$\mathbb{N}^*$	Dimensionality of the document vector
g	Function	Similarity function (e.g., inner product)
s_i	$\mathbb{R}$	Similarity score between query and document x_i
M	$\mathbb{N}^*$	Number of partitions in the document index
$\mathcal{D}_m$	Subset of $\mathcal{D}$	Partition m of the document collection
$\mathbf{r}_m$	$\mathbb{R}^J$	Vector representation of partition $\mathcal{D}_m$
v_m	$\mathbb{R}$	Similarity score between query and partition $\mathcal{D}_m$

Table 5: Notation definitions and explanations

Algorithm 2 Inference with Hildoc

Input: query y, number of partitions to retrieve C, number of documents to retrieve k.

Output: top-k documents

```
1:  $\mathbf{q} \leftarrow f(y)$
2:  for $m = 1$ to $M$ do
3:      $v_m \leftarrow g(\mathbf{q}, \mathbf{r}_m)$
4:  end for
5:  $l_1 \dots l_C \leftarrow top\_ind(\{v_m\}_{m=1}^{M}, C)$ {Obtain the top-C partitions}
6:  $\mathcal{S} \leftarrow \varnothing$
7:  for $i = 1$ to $C$ do
8:      $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{D}_{l_i}$ {Combine all the documents in the top-C partitions}
9:  end for
10: for $i = 1$ to $|\mathcal{S}|$ do
11:     Retrieve the i-th document $x_i$ inside $\mathcal{S}$
12:     Obtain the pre-computed document embedding $\mathbf{d}_i = f(x_i)$
13:     $s_i \leftarrow g(\mathbf{q}, \mathbf{d}_i)$
14: end for
15: $j_1 \dots j_k \leftarrow top\_ind(\{s_i\}_{i=1}^{|\mathcal{S}|}, k)$ {Obtain the top-k documents based on similarity}
16: Return $x_{j_1} \dots x_{j_k}$ from $\mathcal{S}$
```

D Computational Complexity

From Algorithm 1, we can see that the "argsort" step on line 5 requires sorting the Hilbert Curve Representations of N documents. This is the most computationally expensive step in the whole training process, which can take $O(N\log(N))$ in both space and time complexity. However, many efficient sorting algorithms are available, especially the ones that leverages parallel computing, which can robustly sort 10^{12} items or more (O'malley, 2008). In addition, Algorithm 1 only requires three iterations over the documents (line 1, 7, and 14), which takes $O(N)$ time complexity.

In terms of inference, the algorithm needs to iterate through the M partitions and the documents inside the top-C partitions. Due to Hildoc's high Uniformity (Proposition 2), each partition contains maximum $\frac{2N}{M}$ documents. Hence, the retrieved top-C partitions contains $\frac{2NC}{M}$ documents at max. Hence, the overall time complexity for inference is $O(M + \frac{2NC}{M})$, where M and C are configurable during training and inference to manage the trade-off between speed and accuracy.

E Calculation of Hilbert Curve

Efforts to calculate the Hilbert Curve in a computationally efficient and numerically stable manner have been numerous (Skilling, 2004). Typically, Hilbert Curves are computed recursively until a pre-specified order, T, is reached. Additionally, there are several practical implementations available, including the *hilbertcurve* and *numpy-hilbert-curve* libraries in Python.

F Additional Experiment Results

The experimental results on MSMARCO passage dataset which encoded using T5-ANCE embedding model are listed in Table 6. As shown from the

Method	MRR@10	MRR@50	MRR@100	R@1	R@20	R@50
MEVI (C-10)	32.037	32.607	32.637	21.317	58.841	63.232
MEVI (C-100)	35.160	36.070	36.128	22.536	70.558	79.106
MEVI (C-1000)	35.748	36.743	36.815	22.822	72.611	82.427
MEVI Ens (C-10)	35.373	36.357	36.432	22.611	71.660	81.320
MEVI Ens (C-100)	35.620	36.614	36.689	22.730	72.641	82.458
MEVI Ens (C-1000)	35.739	36.725	36.803	22.766	72.882	82.747
Hildoc (C-10)	31.451	32.270	32.313	20.475	62.043	69.548
Hildoc (C-100)	34.421	35.356	35.420	22.170	69.280	78.479
Hildoc (C-1000)	35.362	36.353	36.425	22.622	71.763	81.636
Hildoc Ens (C-10)	34.692	35.604	35.664	22.384	69.253	77.777
Hildoc Ens (C-100)	35.584	36.573	36.642	22.704	72.519	82.258
Hildoc Ens (C-1000)	35.875	36.881	36.957	22.962	73.007	82.919

Table 6: Experimental results on MSMARCO Passage with embedding model T5-ANCE.

table, the performance results show that Hildoc Ens (C-1000) achieved the best performance. Note that, C-1000 denotes that the number of retrieved partitions.

Table 7 lists the performance results of different methods on encoded MSMARCO dataset using AR2 model. As shown in Table 7, Hildoc improved the performance results when combined with MEVI. On MRR metric, Hildoc Ens (C-1000) achieved an improvement of 0.629 to 0.654 over MEVI Ens (C-1000). Likewise, the improvement on Recall was 0.661 to 1.249.

Table 8 shows the performance of the models on natural questions dataset. As listed in the table, Hildoc Ens (C-1000) surpasses all methods on MRR and recall metrics with an improvement of 1.636 to 1.674 and 1.053 to 1.717, respectively.

Table 9 shows the trade-off between the mean inference time and the performance of different methods on the NQ dataset. The ratio between the inference time and the MRR@10 measures the suitability of the method to be utilized in online real applications. Hildoc (C-10) method has the lowest ratio 5.24, achieved performance of 60.193% on MRR@10 with inference time of 315.21 Milliseconds. MEVI (C-10) scored 5.33, the second place, achieving performance of 51.483% on MRR@10 metric in 274.27 Milliseconds. As listed in Table 9, Hildoc (C-1000) achieved performance of 65.404% in 448.85 milliseconds while MEVI(C-1000) achieved higher MRR@10, 64.534%, in higher time, 8429.81 milliseconds. Therefore, Hildoc is more flexible than MEVI in trading-off between efficiency and performance by increasing the number of retrieved clusters which makes it more suitable for online applications.

Method	MRR@10	MRR@50	MRR@100	R@1	R@20	R@50
MEVI (C-10)	33.659	34.190	34.214	22.931	59.873	63.787
MEVI (C-100)	38.114	38.960	39.008	25.016	73.056	81.146
MEVI (C-1000)	39.268	40.201	40.266	25.768	76.334	85.521
MEVI Ens (C-10)	36.996	37.972	38.045	24.194	73.128	82.531
MEVI Ens (C-100)	38.391	39.339	39.402	25.080	74.925	84.522
MEVI Ens (C-1000)	38.995	39.942	40.008	25.335	76.334	85.762
Hildoc (C-10)	33.967	34.747	34.779	22.202	65.384	72.440
Hildoc (C-100)	37.489	38.399	38.457	24.574	72.952	81.761
Hildoc (C-1000)	38.868	39.805	39.871	25.448	75.904	85.050
Hildoc Ens (C-10)	37.802	38.689	38.737	24.962	72.723	81.058
Hildoc Ens (C-100)	39.303	40.261	40.321	25.897	76.814	86.151
Hildoc Ens (C-1000)	39.624	40.595	40.662	25.997	77.487	87.011

Table 7: Experimental results on MSMARCO Passage (Dev) with embedding model AR2.

Method	MRR@10	MRR@50	MRR@100	R@1	R@20	R@50
MEVI (C-10)	51.483	51.789	51.816	45.402	66.399	69.640
MEVI (C-100)	60.449	60.748	60.775	53.407	77.285	80.028
MEVI (C-1000)	64.534	64.850	64.872	56.787	82.881	85.679
MEVI Ens (C-10)	63.551	63.870	63.895	55.789	82.548	85.180
MEVI Ens (C-100)	63.424	63.750	63.772	55.485	82.992	86.399
MEVI Ens (C-1000)	64.641	64.972	64.999	56.620	84.266	87.091
Hildoc (C-10)	60.193	60.505	60.524	52.188	79.141	81.551
Hildoc (C-100)	64.046	64.366	64.389	56.233	82.770	85.734
Hildoc (C-1000)	65.404	65.724	65.748	57.313	84.626	87.202
Hildoc Ens (C-10)	63.806	64.090	64.108	56.233	81.994	84.404
Hildoc Ens (C-100)	65.677	65.986	66.010	57.950	84.681	87.341
Hildoc Ens (C-1000)	66.315	66.612	66.635	58.338	85.679	88.144

Table 8: Experimental results on Natural Questions.

Method	MRR@10	Inference time	Inference time/MRR
MEVI (C-10)	51.483	274.27	5.33
MEVI (C-100)	60.449	1057.00	17.49
MEVI (C-1000)	64.534	8429.81	130.63
MEVI Ens (C-10)	63.551	2819.61	44.37
MEVI Ens (C-100)	63.424	3602.34	56.80
MEVI Ens (C-1000)	64.641	10975.15	169.79
Hildoc (C-10)	60.193	315.21	5.24
Hildoc (C-100)	64.046	356.50	5.57
Hildoc (C-1000)	65.404	448.85	6.86
Hildoc Ens (C-10)	63.806	588.93	9.23
Hildoc Ens (C-100)	65.677	1413.49	21.52
Hildoc Ens (C-1000)	66.315	8878.66	133.89

Table 9: Trade-off between query mean inference time (in milliseconds) and model performance on NQ dataset.