Why These Documents? Explainable Generative Retrieval with Hierarchical Category Paths

Anonymous ACL submission

Abstract

Generative retrieval directly decode a document identifier (i.e., docid) in response to a query, making it impossible to provide users with explanations as an answer for *"why is this document retrieved?"*. To address this limitation, we propose **H**ierarchical Categor**y** **P**ath-**E**nhanced Generative Retrieval (**HYPE**), which enhances explainability by first generating hierarchical category paths step-by-step then decoding docid. By leveraging hierarchical category paths which progress from broader to more specific semantic categories, HYPE can provide detailed explanation for its retrieval decision. For training, HYPE constructs category paths with external high-quality semantic hierarchy, leverages LLM to select appropriate candidate paths for each document, and optimizes the generative retrieval model with path-augmented dataset. During inference, HYPE utilizes path-aware ranking strategy to aggregate diverse topic information, allowing the most relevant documents to be prioritized in the final ranked list of docids. Our extensive experiments demonstrate that HYPE not only offers a high level of explainability but also improves the retrieval performance.

1 Introduction

Information retrieval (IR) systems are essential for helping users find proper information within vast amount of online information. A fundamental task of these systems is document retrieval, which focuses on searching for and ranking documents that are relevant to a given query from a large document corpus. Recently, *generative retrieval* has emerged as a new paradigm in document retrieval. It aims to directly generate document identifier (i.e., docid) for a given query by leveraging pre-trained generative models such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). This paradigm enables end-to-end optimization of the retrieval process, allowing for fine-grained interaction between the input query and docid, and significantly reduces memory usage by leveraging the parametric memory of a single generative model.

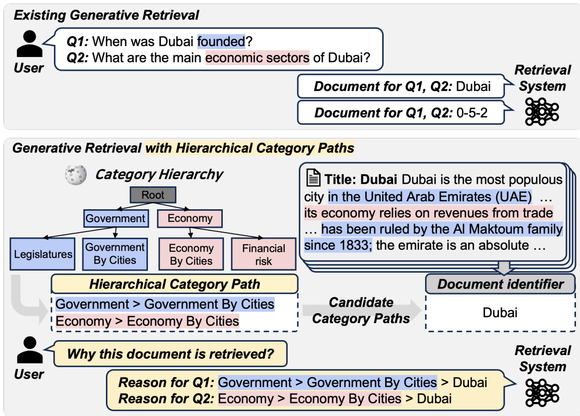


Figure 1: Existing generative retrieval methods fail to explain why specific documents are retrieved, as they directly decode docid (Upper). In contrast, our HYPE provides clear explanations by generating query-related hierarchical category paths leading to the docid (Lower).

Even with these advantages, generative retrieval continues to face the challenge of determining how to construct docid that effectively represent documents. As the docid serves as a representation of the entire document, defining one that accurately encapsulates the document's contents is both crucial and challenging. Existing works on generative retrieval have categorized docid into two types: *semantic* docid and *lexical* docid. A semantic docid represents each document as a series of numbers (e.g., 0-5-2), where each number indicates a cluster index assigned over its dense representation. This dense representation is encoded by a PLM-based encoder (Devlin et al., 2019; Raffel et al., 2020) and clustered using methods such as hierarchical k-means (Tay et al., 2022; Wang et al., 2022) or product quantization (Zhou et al., 2022). On the other hand, lexical docid represents each document as human-readable text, such as titles (Cao et al., 2021), keywords (Zhang et al., 2023; Wang et al.,

2023) and pseudo queries (Tang et al., 2023).

However, both existing approaches still lack explainability, which remains a significant limitation. For instance, in the upper part of Figure 1, two types of queries related to the same document "Dubai", are presented. While the existing retrieval systems may return identifiers of relevant documents such as the lexical docid (i.e., Dubai) or semantic docid (i.e., 0-5-2), they fail to provide an explicit explanation that aligns with the different intention behind each query. **Specifically, they do not clarify why a particular document is retrieved for a specific query and fail to answer the question, "*why is this document retrieved?*".** The lack of explainability in retrieval systems is a critical issue, as it can undermine the reliability of retrieved documents and make it more difficult for users to explore additional information related to a specific query (Anand et al., 2022). To address this aforementioned limitation, our research aims to design a generative retrieval framework that can provide retrieved document with clear and reasonable explanations for a user's query.

In this work, we propose **H**ierarchical Categor**y** **P**ath-**E**nhanced Generative Retrieval (**HYPE**), which enhances explainability by generating hierarchical category paths step-by-step before decoding docid. Motivated by structured document categorization systems, such as Wikipedia category tree or Microsoft Academic taxonomy (Shen et al., 2018), HYPE utilizes hierarchical category paths as explanations, progressing from broad to specific semantic categories. In the lower part of Figure 1, when queries about document "Dubai" are given, HYPE uses category paths like *"Government > Government by cities"* or *"Economy > Economy by cities"* to explain why document "Dubai" is retrieved for each query. This approach 1) enables specific explanations for the document depending on the query by using hierarchical category paths that connect the query and the document, and 2) provides more reasonable and insightful explanation by reflecting the document's semantic structure through a coarse-to-fine manner. Additionally, HYPE 3) can employ effective ranking of the retrieved results by leveraging multiple paths, which helps improve retrieval performance.

Specifically, HYPE consists of the following three steps: 1) constructing category paths based on an external semantic hierarchy and selecting appropriate candidate paths for each document using Large Language Models (LLM), 2) building a path-augmented dataset with candidate paths, and 3) optimizing a model with the path-augmented dataset. During inference phase, HYPE conducts a pseudo-reasoning process[1] by generating the hierarchical category path step-by-step to decode docid, allowing it to serve as an explanation which enhances explainability. Additionally, HYPE employs *path-aware ranking* strategy, which simultaneously considers multiple pseudo-reasoning paths for each query. This strategy helps build a more robust retrieval system by capturing the semantic information of multiple category paths, thereby improving overall retrieval performance.

Our extensive experiments demonstrate that HYPE not only offers a high level of explainability but also improves the retrieval performance in the document retrieval task. Additionally, HYPE can be applied orthogonally to various docid types (e.g., *title*, *keywords*), making it a versatile framework that can be seamlessly integrated into different generative retrieval systems. For reproducibility, our codes are publicly available at the anonymous github repository.[2]

We summarize our contributions as follows:

- We introduce HYPE, an explainable generative retrieval framework that generates query-specific hierarchical category paths for relevant documents before decoding their docid. These category path enables the retrieval system to provide users explanations of document retrieval.
- We propose a new ranking strategy called *path-aware ranking*, which considers multiple category paths simultaneously to determine the final ranked list of docids.
- We empirically show that HYPE improves both the explainability and accuracy of generative retrieval across various docid types, making it adaptable and easily integrable into different generative retrieval systems.

2 Preliminaries

In this section, we formally define the task of generative retrieval and explain its overall process and relevant techniques.

2.1 Task Formulation

Given a corpus $\mathcal{C} = \{D_1, D_2, \ldots, D_n\}$ where D represents a document, generative retrieval aims to autoregressively generate the document identifier

[1]We describe this term in Appendix A.2.

[2]https://anonymous.4open.science/r/HyPE-1B74

(i.e. docid) of the relevant document for a given query. To this end, the model is optimized for **indexing task** and **retrieval task**. The indexing task involves taking a document as the input and generating the corresponding docid, described by

$$\mathcal{M}^{\theta}(d \mid D) = \prod_{t=1}^{n} \mathcal{M}^{\theta}(d_t \mid D, d_{<t}), \tag{1}$$

where $\mathcal{M}^{\theta}$ is a generative model, D is a document, d is the target docid, and n is the token length of the target docid. The retrieval task focuses on processing a query as the input and generating the docid of a relevant document, described as follows:

$$\mathcal{M}^{\theta}(d \mid q) = \prod_{t=1}^{n} \mathcal{M}^{\theta}(d_t \mid q, d_{<t}), \tag{2}$$

where q is a query. In performing the aforementioned two tasks, it is crucial to address two key aspects: 1) effectively represent the long document D and 2) construct the docid d that captures the overall semantic information of the document.

During inference, given an input query q, the model produces a top-K ranked list of docids that have the largest likelihoods $\mathcal{M}^{\theta}(d \mid q)$. To ensure the generation of valid docids, the model employs constrained decoding, which mostly uses constrained beam search (Cao et al., 2021).

2.2 Document Representation and Identifier

Document representation. For the indexing task, each document is used as the input. This makes it crucial to define effective input representations of the long document while preserving as much of its information as possible within the context length of the language model. The primary approaches to effectively representing documents are FirstP (Tay et al., 2022) and Document as Query (DaQ) (Wang et al., 2022). FirstP uses only the first k tokens from the beginning of the document, while DaQ randomly extracts chunks from the document.

Document identifier. To ensure that docid effectively encodes semantic information of document, a variety of approaches have been proposed. Docid can be broadly categorized into *semantic docid* and *lexical docid*. Semantic docid represents each document as a series of numbers, where each number corresponds to a cluster index derived from the document's dense representation. This dense representation is encoded by a PLM-based encoder (Devlin et al., 2019) and mapped to discrete cluster indices using methods such as hierarchical k-means (Tay et al., 2022; Wang et al., 2022) or product quantization (Zhou et al., 2022). Lexical docid is a textual format designed to effectively convey the semantic content of a document. It can be constructed using various forms, such as the document's title (Cao et al., 2021), substrings (Bevilacqua et al., 2022), keywords (Zhang et al., 2023; Lee et al., 2023; Wang et al., 2023), URL (Zhou et al., 2022), and pseudo query (Tang et al., 2023). Title and URL are used as docid directly from the dataset. Substrings are generated by the retrieval model using an FM index (Ferragina and Manzini, 2000), which creates specific n-grams within the document for retrieval. Keywords are extracted from the document using methods such as TF-IDF (Robertson and Walker, 1997), BM25 (Robertson and Zaragoza, 2009), or pre-trained language models (PLMs). Pseudo query is generated using query generation models, such as docT5query (Nogueira and Lin, 2020), which is then utilized as the docid.

2.3 Optimization and Inference

Optimization via multi-task learning. Given a training dataset that consists of (query, document, docid), denoted by $\mathcal{X} = \{(q, D, d)\}$, the model is trained for both the indexing and retrieval tasks, maximizing the likelihoods in Equations (1) and (2), respectively:

$$\max_{\theta} \sum_{(q,D,d)\in\mathcal{X}} \mathcal{M}^{\theta}(d \mid D) + \mathcal{M}^{\theta}(d \mid q) \tag{3}$$

Indexing with synthetic query. In indexing task, documents are long and contain extensive information; however, in retrieval task, queries are relatively short and request specific information. To bridge this discrepancy, recent studies (Zhuang et al., 2023; Wang et al., 2022; Sun et al., 2023) have tried to integrate synthetic queries, generated by query generation models (Nogueira and Lin, 2020), into the training phase. The synthetic queries improve the retrieval performance of generative retrieval models by effectively reducing the gap between queries and documents. Note that these synthetic queries are treated as alternative document representation, similar to FirstP and DaQ mentioned in 2.2, and are used as input for the indexing task (Zhuang et al., 2023; Sun et al., 2023).

3 Proposed Method

In this section, we present **H**ierarchical categor**y** **P**ath-**E**nhanced generative retrieval (**HyPE**),

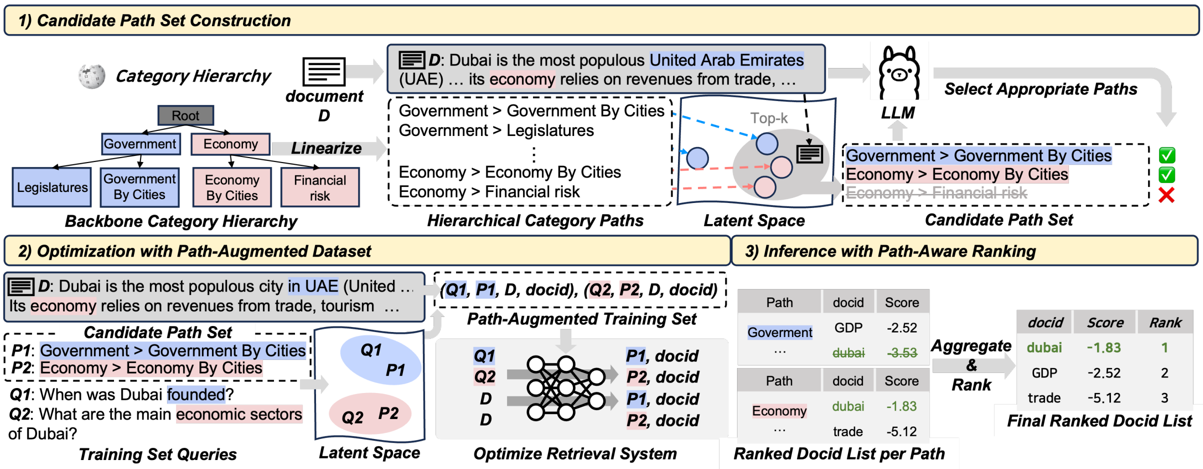


Figure 2: Overview of HYPE framework. (1) HYPE constructs category paths using an external high-quality semantic hierarchy and employs LLM to select appropriate candidate paths for each document. (2) Then, HYPE links queries to the paths based on semantic relevance to construct path-augmented training set, and uses this to optimize the retrieval system. (3) During inference, HYPE employs path-aware ranking strategy to determine the final docid ranking by considering multiple paths.

which improves explainability by generating hierarchical category paths step-by-step before decoding docid. The overall framework is shown in Figure 2.

3.1 Candidate Path Set Construction

The first step of our HYPE framework is to construct a set of candidate hierarchical category paths for each document. To ensure explainability, these paths should satisfy the following criteria: *Semantic Hierarchy*, *Generalizability*, and *Specificity* (see Appendix A.3 for details). To achieve this, we first construct the high-quality backbone hierarchy for category paths. Then, for each document, we (1) filter out category paths based on semantic similarity calculated by a pre-trained text encoder, and (2) select several category paths that comprehensively represent the content of the document while specifically addressing certain topics within the document by the help of reasoning capabilities of LLM.

Hierarchical category path collection. In the open-domain retrieval task, the category (or topic) hierarchy must encompass both a broad range of domain categories (i.e. width of tree) and sufficient semantic granularity (i.e. depth of tree) to ensure comprehensive and accurate retrieval system. To this end, we leverage Wikipedia's category tree as our backbone hierarchy of categories, setting the *Main Topic classification* category as the root node of the hierarchy. This hierarchy is specifically designed to systematically categorize "real-world wikipedia documents", which cover a wide range of domains and provide specific and detailed semantic information. Considering the vast and complex nature of Wikipedia's category tree, we limit the scraping process to a depth of four to construct our backbone hierarchy. Then, we linearize all the paths within the hierarchy and convert them into a sequence of strings, thereby enabling more efficient processing and manipulation. The entire set of linearized category paths is denoted by $\mathcal{P}$. The statistics of collected hierarchical category paths are presented in Appendix A.3.

Candidate path set construction. Subsequently, we utilize the knowledge of LLM to assign appropriate category paths to each document within the corpus. However, due to the context length of LLM, it is impossible to input all possible paths within the category hierarchy (collected in Section 3.1). Thus, we first filter out path set for each document D by leveraging a bi-encoder. The pre-candidate path set $\hat{\mathcal{P}}_D$ is obtained as follows:

$$\hat{\mathcal{P}}_D = \underset{p \in \mathcal{P}}{\text{argTop-}k} \ \text{sim}(E(D), E(p)), \quad (4)$$

where $E(\cdot)$ is the encoder, $\text{sim}(\cdot)$ is a cosine similarity, and k is the number of pre-candidate paths for each document. Then, given the document D and its pre-candidate path set $\hat{\mathcal{P}}_D$, we leverage LLM[3] to generate the final path set $\mathcal{P}_D$, selecting up to three paths that best represent the document.

3.2 Optimization with category path

The second step is to augment the training set $\mathcal{X}$ with path, building a path-augmented training set

[3]We use Llama-3-8B-Instruct (Dubey et al., 2024) as LLM.

$\mathcal{X}^+ = \{(q, p^q, D, d)\}$. To achieve this, we first (1) link each query to one of the document's candidate paths based on semantic similarity computed by pre-trained encoder, and then (2) utilize the resulting query-path pairs together with the document-path pairs to optimize the retrieval model.

Linking Path with Query. Using the candidate path set for each document, we build a *path augmented training set* $\mathcal{X}^+$. For each query-document pair in the training set $(q, D, d) \in \mathcal{X}$, we link the query q to its most relevant path among the paths in the document's candidate path set P_D. This linking can be described as follows:

$$p^q = \underset{p \in \mathcal{P}_D}{\operatorname{argmax}} \ \operatorname{sim}(E(q), E(p)), \tag{5}$$

where p^q is the path linked to the query q. This process is then applied to all queries in the training set. In the end, we construct the path-augmented training set, denoted by $\mathcal{X}^+ = \{(q, p^q, D, d)\}$.

Optimization. By leveraging the path-augmented training set $\mathcal{X}^+$, we train our model $\mathcal{M}^\theta$ on both indexing and retrieval tasks, as described in 2.1. Our optimization follows the same strategy as standard generative retrieval in 2.1, with the only difference being the addition of path information as follows:

$$\max_\theta \sum \mathcal{M}^\theta(p^q, d \mid D) + \mathcal{M}^\theta(p^q, d \mid q) \tag{6}$$

3.3 Inference with Path-Aware Ranking

During inference, HYPE generates the final ranked list of docids through two stages: 1) *path generation stage* and 2) *docid decoding stage*. First, in the path generation stage, our model $\mathcal{M}^\theta$ generates up to K_p hierarchical category paths, each of which is denoted by p_j for $j = 1, \ldots, K_p$, by using beam search; these are query-specific hierarchical category paths that encapsulate various topics related to the given query. Next, in the docid decoding stage, the model uses each generated hierarchical category path as the decoder's input context and then applies constrained beam search to decode m docids. For each path p_j, the model outputs m number of docid-score pairs as follows:

$$Y_j = \{(d_i, s_i) \sim \mathcal{M}^\theta(\cdot \mid q, p_j)\}_{i=1}^m, \tag{7}$$

where s_i represents the score for the docid d_i conditioned on the category path p_j. The remaining process is to aggregate K_p number of docid-score pair sets for making the final ranked list of docids. At this point, we remain only unique docid with the highest score, resulting in $\tilde{Y}$.

$$\tilde{Y} = \left\{(d, s) \mid s = \max\{s' | (d, s') \in Y_j\}, \forall (d, s) \in \cup_{j=1}^{K_p} Y_j\right\} \tag{8}$$

From the set of unique docid-score pairs, we obtain the final ranked list by sorting their scores in descending order, $Y_{\text{final}} = \text{sort}(\tilde{Y})$. By utilizing *path-aware ranking* strategy, HYPE can effectively capture the semantic information of an input query from multiple category paths, leading to improved retrieval performance.

4 Experiments

In this section, we design and conduct our experiments to answer the following research questions:

- **RQ1:** Can HYPE improve retrieval accuracy?
- **RQ2:** Can hierarchical category paths in HYPE serve as effective explanations for retrieval?
- **RQ3:** Can explanations of HYPE help real-world users in search systems?

4.1 Experimental Settings

Dataset. We conduct our experiments on two datasets, **NQ320K** (Kwiatkowski et al., 2019) and **MS MARCO** (Nguyen et al., 2016), which have been widely utilized in previous works (Tay et al., 2022; Wang et al., 2022). For NQ320K, we divide the test set into two subsets, *seen* and *unseen*, following the setup in (Wang et al., 2022; Sun et al., 2023), where the *seen* test includes queries whose annotated target documents are present in the training set, and the *unseen* test consists of queries with no labeled documents in the training set. More details are provided in Appendix A.4.

Evaluation Metrics. We report Recall and Mean Reciprocal Rank (MRR) for NQ320K and MS MARCO. For NQ320K, we use Recall@{1, 10, 100} and MRR@100. For MS MARCO, we use Recall@{1, 10, 100} and MRR@10 as done in previous works (Sun et al., 2023; Wang et al., 2023).

Baselines. To validate the effectiveness of HYPE across diverse generative retrieval settings, we conduct experiments on four representative docid types, introduced in Section 2.2, as our baseline.

- **Title docid** uses a document's title as docid. For documents without a title, we use the first 16 tokens of the document as a title, following the approach used in (Sun et al., 2023).

Method	Full test				Seen test				Unseen test			
	R@1	R@10	R@100	M@100	R@1	R@10	R@100	M@100	R@1	R@10	R@100	M@100
Title docid	62.2	78.7	89.3	68.6	64.8	81.5	90.1	71.2	53.1	68.9	80.4	59.3
+ HYPE	63.6*	83.5*	90.1*	71.0*	66.4*	86.3*	92.6*	73.9*	53.7*	73.6*	81.7*	61.0*
Improvement	**+2.3%**	**+6.1%**	**+2.5%**	**+3.5%**	**+2.5%**	**+5.9%**	**+2.8%**	**+3.8%**	**+1.1%**	**+6.8%**	**+1.6%**	**+2.9%**
Keyword docid	61.8	77.1	85.5	67.6	67.3	82.3	89.9	73.0	43.0	59.0	70.4	48.8
+ HYPE	60.7	79.1*	86.2*	67.6	66.6	84.6*	90.7*	73.4*	40.1	60.2*	70.6*	47.5
Improvement	-1.8%	**+2.6%**	**+0.8%**	+0.0%	-1.0%	**+2.8%**	**+0.9%**	**+0.5%**	-6.7%	**+2.0%**	**+0.3%**	-2.7%
Summary docid	60.9	78.8	84.1	67.6	65.7	84.1	88.6	72.6	44.0	60.5	68.5	50.1
+ HYPE	61.5*	79.6*	85.2*	68.3*	66.3*	84.6*	89.8*	73.2*	44.8*	62.2*	69.4*	51.3*
Improvement	**+1.0%**	**+1.0%**	**+1.3%**	**+1.0%**	**+0.9%**	**+0.6%**	**+1.4%**	**+0.8%**	**+1.8%**	**+2.8%**	**+1.3%**	**+2.4%**
Atomic docid	65.3	83.5	89.3	72.2	70.2	88.3	93.5	77.2	48.6	66.8	74.9	55.0
+ HYPE	64.5	84.2*	90.2*	71.9	69.5	88.6*	93.8*	76.8	47.2	68.7*	77.6*	55.0
Improvement	-1.2%	**+0.8%**	**+1.0%**	-0.4%	-1.0%	**+0.3%**	**+0.3%**	-0.5%	-2.9%	**+2.8%**	**+3.6%**	+0.0%

Table 1: Retrieval accuracy of baselines and our HYPE framework on the NQ320K. $*$ denotes the statistical significance on paired t-test $p < 0.05$.

Method	R@1	R@10	R@100	M@10
Keyword docid	31.7	61.2	77.2	41.0
+ HYPE	32.2*	62.7*	78.5*	41.9*
Improvement	**+1.6%**	**+2.5%**	**+1.7%**	**+2.2%**
Summary docid	28.1	55.5	71.5	36.8
+ HYPE	28.4*	57.5*	73.1*	37.8*
Improvement	**+1.1%**	**+3.6%**	**+2.2%**	**+2.7%**
Atomic docid	43.9	73.6	85.6	53.8
+ HYPE	44.9*	74.6*	87.1*	54.7*
Improvement	**+2.3%**	**+1.4%**	**+1.8%**	**+1.7%**

Table 2: Retrieval accuracy of baselines and HYPE on the MS MARCO. $*$ denotes the statistical significance on paired t-test $p < 0.05$.

- **Keyword docid** uses a sequence of keywords as docid that effectively represent the document. For NQ320K, we use 3 keywords, while for MS MARCO, we extract 5 keywords.
- **Summary docid** uses the document summary as docid. Although it has not been attempted before, a similar structure using substrings is employed in (Bevilacqua et al., 2022).
- **Atomic docid** uses a unique arbitrary integer as docid. We assign each document a integer and generates a corresponding new token for it.

We intentionally do not consider semantic docids (+HYPE) in our experiments. This is because semantic docids are constructed based on techniques such as hierarchical clustering, and thus inherently embed a semantic structure. Given that these structures are already formed in a coarse-to-fine manner, prepending hierarchical category paths to them can contradict the coarse-to-fine principle.

Furthermore, existing generative methods employ various architectures and optimization techniques, which may introduce additional factors affecting performance. **To specifically assess the impact of HYPE, we adopt the basic form of generative retrieval described in Section 2 as our baseline.** This approach ensures a direct comparison between plain docids and those enhanced with HYPE, isolating the effects of HYPE itself from other architectural or optimization differences. For more details, please refer to the Appendix A.7.

4.1.1 Implementation Details

We use T5-base (Raffel et al., 2020) as our backbone model. For the input of the indexing task, we utilize the FirstP approach as our document representations and five synthetic queries. (Section 2.2). During the inference of HYPE, we generate three category paths (i.e., $K_p = 3$), and for the docid decoding stage, we use constrained beam search with a beam size of 100 (i.e., $m = 100$). More details about this part are provided in Appendix A.7.

4.2 HYPE improves retrieval accuracy (RQ1)

Table 1 shows retrieval accuracy of various docid types with HYPE on NQ320K. Overall, HYPE consistently improves retrieval accuracy across all docid types in both *seen test* and *unseen test*. This demonstrates that **HYPE's hierarchical category paths can be orthogonally applied to enhance retrieval accuracy across different docid types, suggesting that integrating these paths into existing generative retrieval methods can further improve performance.** While HYPE can be applied to all docid types effectively, the experimental results show that *title* docid yields the most significant performance improvement when HYPE is applied. Our paths, serve as a pseudo-reasoning, allowing the model to navigate step-by-step through various semantic hierarchical categories before arriving docid. Since titles are concise and inherently reflect a structured overview of a document, they aligns well with the HYPE's hierarchical category paths, further enhancing retrieval accuracy.

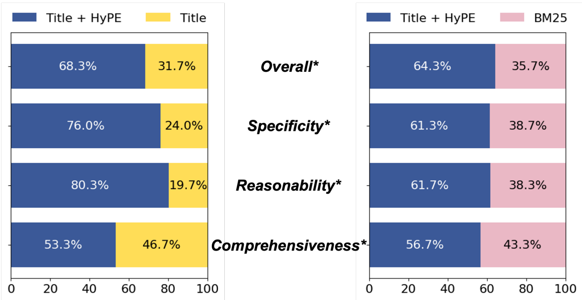


Figure 3: Human evaluation of pairwise quality comparisons for retrieval explanations, generated by HYPE and baseline models.

Additionally, to investigate whether our hierarchical category paths perform effectively on documents beyond Wikipedia, we conduct experiment with MS MARCO. Table 2 shows that HYPE consistently improves retrieval accuracy on MS MARCO as well. Although the hierarchical category paths are constructed using Wikipedia category tree as the backbone, **the consistent performance gains on MS MARCO emphasize the robustness and generalizability of HYPE.** These findings suggest that HYPE can be widely applied to datasets across various domains in the future.

4.3 Hierarchical category paths serve as effective retrieval explanations (RQ2)

We evaluate the explanatory quality of the hierarchical category paths of HYPE through a human evaluation conducted via Amazon Mechanical Turk (AMT). We ask three human judges per sample to compare the quality of the explanations based on four distinct criterias: *overall*, *specificity*, *reasonability* and *comprehensiveness*. Detailed descriptions of the evaluation criteria and experimental baselines are provided in Appendix A.5.

In Figure 3, HYPE outperforms both the title docid baseline and BM25 across all criteria, receiving high scores for its *overall* explanation of the retrieval process. Specifically, HYPE shows substantial margin of superiority in terms of *specificity* and *reasonability*. This demonstrates that HYPE provides clearer explanations of retrieval process, as well as more logical and reasonable explanation. Furthermore, HYPE beats other baselines in *comprehensiveness*, indicating that its hierarchical category path is effective in explaining not only narrow, specific details but also broader semantic information. **These results highlight that HYPE's pseudo-reasoning, which utilizes hierarchical category paths, provides users with a effective explanation of the retrieval process.**

Baseline	R@1	M@5	Conf.
Title Docid	19.7	47.9	4.0
+ HYPE	24.3	52.8	4.5
Improvement	23.7%	10.4%	12.0%

Table 3: Human reranking performance with and without category paths on NQ320K dev set pairs where the model retrieves the gold document in the top 5.

4.4 HYPE guides users in making better search decision by explanations (RQ3)

In real-world search systems, users are typically provided only with the document title and the first few lines when deciding which result to open. We investigate whether explanations of HYPE can help users effectively identify relevant documents in such real-world settings. To this end, we conduct a human reranking experiment via AMT using the NQ320K dev set. Specifically, human judges rerank the top-5 retrieved results by relevance and rate their confidence (1–5) under two settings: title only, and title with category path. With human-reranking results, we measure performance with Recall@1, MRR@5 and *Confidence*. Details of the evaluation setup are provided in Appendix A.6.

Table 3 shows that offering hierarchical category paths improve human reranking accuracy, with Recall@1 improving by 23.7% and MRR@5 by 10.4%. This shows that the hierarchical category paths, used as explanations in HYPE, help real-world users better select relevant documents. Additionally, *Confidence* also improves by 12.0%. **These results demonstrate that explanations of HYPE provide users with clarity and guidance, enabling not only more accurate selections but also more confident decisions during search.**

5 Analysis

Case Study. Table 4 illustrates HYPE's explanations in cases where a single document is annotated with multiple queries on different topics. For the query "*the core of the sun in which the sun's thermonuclear energy is produced*", the model generates paths related to the universe and energy conversion, clearly explaining the thematic relevance between the query and the document. However, for another query, "*what stage of the star life cycle is the sun in*", it generates a path related to stellar evolution, which is different from the previously generated path but relevant to the query. **This shows that HYPE can provide effective explanations to users by tailoring them to each query.**

Document	Generated Category Paths for Each Query
Title: Sun The Sun is the star at the center of the Solar System. . . . The core is the only region of the Sun that produces an appreciable amount of thermal energy through fusion; . . . The Sun is about halfway through its main-sequence stage, during which nuclear fusion reactions in its core fuse hydrogen into helium.	**Query 1**: the core of the sun in which the sun's thermonuclear energy is produced takes up about **Generated Category Path**: universe > energy > energy conversion **Query 2**: what stage of the star life cycle is the sun in **Generated Category Path**: nature > evolution > stellar evolution

Table 4: Example of the document annotated for multiple queries in the NQ320K dev set. The generative retrieval model with HyPE generates query-specific category paths based on the topics of the document associated with each query, explaining why the document is retrieved for the particular query.

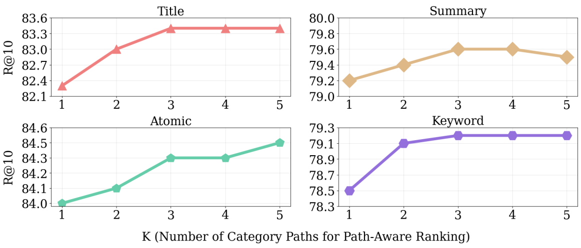


Figure 4: Performance changes of HyPE. The number of decoded category paths to obtain a ranked docid list.

Docid Type	Docid Only	Docid + HyPE
Summary	0.8127s	0.9134s
Keyword	1.0389s	1.1402s

Table 5: Average inference time per instance for decoding only docid vs decoding both docid and a single path.

Analysis of Path-Aware Ranking. To validate the effectiveness of *path-aware ranking* strategy, we analyze the performance changes in retrieval accuracy with respect to the number of hierarchical category paths considered by HyPE. Figure 4 presents the analysis results, showing that retrieval accuracy improves as the number of paths increases across all baselines. Notably, there is a clear performance gap between the setting without path-aware ranking strategy (i.e., $K = 1$) and with path-aware ranking strategy (i.e., $K > 2$). These results indicate that considering multiple paths through the path-aware ranking strategy allows the most relevant docids to be prioritized in the final ranked list, thereby enhancing retrieval accuracy. However, we observe that using too many paths eventually leads to a plateau in performance improvement. Beyond a certain threshold, additional paths tend to introduce noise or increase unnecessary complexity. Consequently, using three paths achieves optimal retrieval accuracy for most docid types.

Analysis of Efficiency. Providing explanations in the context of generative retrieval inherently increases inference cost, as it involves additional explanation generation beyond the decoding docids alone. Considering this, we conduct additional experiments to analyze the impact of HyPE's path generation stage on inference cost. Table 5 compares the average inference time per instance for decoding only docids and decoding docids with HyPE's path generation stage. Details of the analysis setup are provided in Appendix A.8. Overall, when applying HyPE, the inference time increases slightly compared to decoding only docids. Nevertheless, the hierarchical category path employed by **HyPE effectively enhances explainability and retrieval accuracy** by providing a structured and step-by-step way to convey the connection between queries and retrieved documents, **while minimizing the additional computational cost inherently involved in the explanation generation process.**

6 Related Work

Generative retrieval leverages a single pre-trained generative model, such as T5 (Raffel et al., 2020) and BART (Lewis et al., 2020), to directly generate document identifier (docid) relevant to the query, enabling end-to-end optimization of the retrieval process (Tay et al., 2022; Wang et al., 2022; Sun et al., 2023; Wang et al., 2023; Zhang et al., 2023; Zhou et al., 2022; Lee et al., 2023). Additionally, it reduces reliance on external indexing, lowering the system's demand for storage resources. However, existing generative retrieval methods directly generate the docid for a user's query, making it difficult to fully understand why the document is retrieved.

7 Conclusion

In this paper, we propose HyPE, a framework designed to enhance the explainability of document retrieval by utilizing hierarchical category paths. Our experiments demonstrate that HyPE not only enhances overall retrieval performance but also helps users make more accurate decisions during search by providing effective explanations. We hope our research paves the way for meaningful progress in the development of retrieval systems.

Limitations

Despite the promising results and contributions of HyPE, our work has three key limitations stemming from computational costs and budget constraints. First, we do not experiment with alternative backbone hierarchies beyond Wikipedia's category tree. While it is possible that domain-specific taxonomies may further improve retrieval performance in specialized settings, we consider Wikipedia's broad and deep hierarchy sufficient for general-purpose document retrieval. Please refer to Appendix A.3 for further discussion. Second, due to cost and scalability constraints, we do not conduct human evaluations to assess how different path depths affect the quality of the explanation. Instead, we provide a limited analysis of explainability with respect to path depth using STS score in the Appendix A.1. Third, we evaluate HyPE using a basic generative retrieval setup (Section 2) to isolate its effect. We do not incorporate advanced optimization techniques or architectures from recent works, which may further improve performance of HyPE.

Ethical Statement

This study strictly adhered to ethical guidelines throughout the human evaluation and data usage process. All content used in the human evaluation and human reranking—including NQ320K and Wikipedia documents—was publicly accessible and did not involve any private or proprietary data. We did not obtain IRB approval for our study, following precedents set by prior work (Kim et al., 2023; Kang et al., 2024a) which conducted similar human evaluations without IRB oversight. We ensure that no ethical concerns would arise during the evaluation. The evaluation and reranking were conducted on Amazon Mechanical Turk (AMT), where all participation was anonymous and no personal information was collected at any stage. For human evaluation, we hire three different judges per instance from Amazon Mechanical Turk and guarantee fair compensation for each judge. We pay $0.15 for each unit task. Human judges were fully informed about the task's purpose, procedure, and estimated time requirement before beginning the task. Additionally, all examples were screened to exclude offensive, hateful, or sensitive content and were limited to socially and culturally neutral topics. All datasets used in this study are publicly available and appropriately licensed. Specifically, the NQ dataset is distributed under the Apache 2.0 license, and the MS MARCO dataset is released under the MIT license.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Avishek Anand, Lijun Lyu, Maximilian Idahl, Yumeng Wang, Jonas Wallat, and Zijian Zhang. 2022. Explainable information retrieval: A survey. *arXiv preprint arXiv:2211.02405*.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick S. H. Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. In *NeurIPS*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *ICLR*.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

P. Ferragina and G. Manzini. 2000. Opportunistic data structures with applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 390–398.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Dongjin Kang, Sunghwan Kim, Taeyoon Kwon, Seungjun Moon, Hyunsouk Cho, Youngjae Yu, Dongha Lee, and Jinyoung Yeo. 2024a. Can large language models be good emotional supporter? mitigating preference bias on emotional support conversation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 15232–15261, Bangkok, Thailand. Association for Computational Linguistics.

SeongKu Kang, Yunyi Zhang, Pengcheng Jiang, Dongha Lee, Jiawei Han, and Hwanjo Yu. 2024b. Taxonomy-guided semantic indexing for academic paper search. *ArXiv*, abs/2410.19218.

Hyunwoo Kim, Jack Hessel, Liwei Jiang, Peter West, Ximing Lu, Youngjae Yu, Pei Zhou, Ronan Bras, Malihe Alikhani, Gunhee Kim, Maarten Sap, and Yejin Choi. 2023. SODA: Million-scale dialogue distillation with social commonsense contextualization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12930–12949, Singapore. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. 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. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Dongha Lee, Jiaming Shen, SeongKu Kang, Susik Yoon, Jiawei Han, and Hwanjo Yu. 2022. Taxocom: Topic taxonomy completion with hierarchical discovery of novel topic clusters. *Proceedings of the ACM Web Conference 2022*.

Sangam Lee, Ryang Heo, SeongKu Kang, and Dongha Lee. 2025. Imagine all the relevance: Scenario-profiled indexing with knowledge expansion for dense retrieval. *ArXiv*, abs/2503.23033.

Sunkyung Lee, Minjin Choi, and Jongwuk Lee. 2023. Glen: Generative retrieval via lexical index learning. In *Conference on Empirical Methods in Natural Language Processing*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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 *NeurIPS*.

Rodrigo Nogueira and Jimmy Lin. 2020. From doc2query to doctttttquery. *Online preprint*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

S. E. Robertson and S. Walker. 1997. On relevance weights with little relevance information. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '97, page 16–24, New York, NY, USA. Association for Computing Machinery.

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

Zhihong Shen, Hao Ma, and Kuansan Wang. 2018. A web-scale system for scientific knowledge exploration. In *Proceedings of ACL 2018, System Demonstrations*, pages 87–92, Melbourne, Australia. Association for Computational Linguistics.

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2023. Learning to tokenize for generative retrieval. *CoRR*.

Yubao Tang, Ruqing Zhang, J. Guo, Jiangui Chen, Zuowei Zhu, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2023. Semantic-enhanced differentiable search index inspired by learning strategies. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Prakash Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. In *NeurIPS*.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A neural corpus indexer for document retrieval. In *NeurIPS*.

Zihan Wang, Yujia Zhou, Yiteng Tu, and Zhicheng Dou. 2023. Novo: Learnable and interpretable document identifiers for model-based ir. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 2656–2665, New York, NY, USA. Association for Computing Machinery.

Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian M. Sadler, Michelle T. Vanni, and Jiawei Han. 2018. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, and Zhao Cao. 2023. Generative retrieval via term set generation. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. *CoRR*.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2023. Bridging the gap between indexing and retrieval for differentiable search index with query generation. In *Gen-IR@SIGIR*.

A Appendix

A.1 Quantitative Analysis of Explainability

We quantitatively evaluate whether HyPE's hierarchical category path provides a valid explanation by effectively capturing the semantic relationship between the query and the document. To this end, we use a semantic textual similarity (STS) model (Agirre et al., 2012)[4] to measure the semantic relevance between two sentences, evaluating the semantic relevance between the query and explanation, as well as between the document and explanation. Specifically, for each baseline, we use the model output as an explanation and calculate the STS scores for both the query-explanation and document-explanation pairs. We then compute the geometric mean of these two scores to evaluate how effectively the explanation captures the relationship between the query and the document. To further analyze the role of hierarchical category paths in explainability, we consider how varying the maximum level of the paths impacts semantic relevance. As mentioned in Section 3.1, HyPE basically leverages Level 4 paths, but we also experiment with varying the maximum level (e.g., Level 2, Level 3) to examine how the maximum level of paths influences the explainability of the query-document relationship. In addition, we also include BM25 as a baseline, which is capable of providing explanations for its retrieval results. For the explanation of BM25, we consider the top-3 terms that have the highest BM25 scores calculated between a given query and a document.

As shown in Table 6, applying HyPE improves overall semantic relevance across all baselines. This indicates that HyPE's category path effectively captures and explains the relationship between the query and the document. We note that HyPE achieves higher overall relevance than the term-matching method (i.e., BM25), further proving the validity of the HyPE's category path as an explanation. Moreover, maximum level of hierarchical category path significantly influences overall semantic relevance. Specifically, paths with fewer levels than the default level (level 4) fail to capture sufficient semantic relevance between the query and the document, resulting in limited explainability. These results demonstrate that for category paths to effectively serve as explanations, they must achieve *specificity* necessary to sufficiently explain specific and detailed semantic information, as mentioned in Section 3.1.

[4]We use sentence-transformers/roberta-base-nli-stsb-mean-tokens as STS model

Baseline	Semantic Relevance		
	Query	Document	Overall
Title Docid	0.52	0.46	0.48
+ HyPE (Level 2)	0.49	0.51	0.49
+ HyPE (Level 3)	0.49	0.54	0.50
+ HyPE	0.50	0.56	0.52
Keyword Docid	0.42	0.54	0.47
+ HyPE (Level 2)	0.41	0.56	0.47
+ HyPE (Level 3)	0.41	0.57	0.47
+ HyPE	0.43	0.58	0.49
Summary Docid	0.46	0.69	0.55
+ HyPE (Level 2)	0.45	0.70	0.55
+ HyPE (Level 3)	0.45	0.70	0.55
+ HyPE	0.45	**0.71**	**0.57**
BM25	**0.56**	0.31	0.42

Table 6: Semantic relevance between query/explanation and document/explanation on 1,000 NQ320K dev set pairs where each baseline successfully retrieves the relevant document at rank 1.

A.2 Pseudo-Reasoning

Generating the hierarchical path resembles step-by-step reasoning. However, unlike natural language-based reasoning in LLM, we use the term "pseudo-reasoning" because the path structure is more akin to pseudo-code.

A.3 Backbone category hierarchy

Criteria for Selecting the Backbone. To address the criteria mentioned in Section 3.1—Semantic Hierarchy, Generalizability, and Specificity—we utilize Wikipedia's category tree as the foundation for our hierarchical structure, designating the Main Topic classification category as the root node of the hierarchy.

- *Semantic Hierarchy*: Are they semantically hierarchical, allowing step-by-step progression in the generation process to clearly represent a specific semantic level?
- *Generalizability*: Are they able to provide semantic information across a wide range of domains?
- *Specificity*: Are they capable of sufficiently explaining specific and detailed information?

Level 1	Level 2	Level 3	Level 4	Total
40	1,330	13,383	95,240	109,993

Table 7: Statistics of the used category hierarchy, showing the number of nodes at each level (or depth).

Wikipedia category tree Overview. Wikipedia's category tree consists of 40 nodes at level 1, covering broad categories such as *Business, Sports, Science, Philosophy, Language, Health, Government, Culture*, and others. This feature of encompassing a wide range of fields ensures that Wikipedia's category tree satisfies the criterion of *Generalizability*, as it can be applied across various domains. Moreover, these broad categories are further subdivided into increasingly specific subcategories as the level increases. For instance, level 1 *Science* is divided into major subcategories such as *Branches of Science, Scientists*, and *History of Science* at level 2. Among these, *Branches of Science* is further refined into *Applied Science, Formal Science*, and *Social Science* at level 3, which are then expanded into even more specific subcategories like *Computer Science, Agronomy, Metrology*, and *Bioinformatics* at level 4. As the levels progress, the structure captures increasingly detailed semantic information, effectively fulfilling the criterion of *Specificity*. Additionally, the broad-to-specific hierarchical structure of Wikipedia's category tree naturally achieves *Semantic Hierarchy*.

Implementation Details for Path. To utilize Wikipedia's category tree, we employed Selenium[5] to recursively scrape the Wikipedia and extract the Wikipedia category tree. When linearizing the category hierarchy into a hierarchical category path, each category is connected using the delimiter >. The delimiter > is chosen among several candidate delimiters because it showed the highest semantic similarity to the natural language sentence "*the right category is included in the left category*", as measured by Sentence-T5.

Scalability of Our Backbone Hierarchy. We believe that Wikipedia's category tree will function effectively in most document retrieval scenarios. This taxonomy was specifically designed to systematically categorize real Wikipedia documents, which cover a wide range of domains and knowledge. **Its broad and deep structure ensures that it can encompass diverse domains effectively, making it a strong backbone hierarchy for general-purpose retrieval systems.**

Adaptability of HyPE. However, we acknowledge that in more specialized domains—such as expert-driven fields like medicine, law, or scientific literature—the Wikipedia-based hierarchy may not fully capture domain-specific semantics or categorization needs. In such cases, the backbone hierarchy may need to be replaced or augmented with a domain-specific taxonomy better suited to the task. **We note that HyPE is compatible with this setting: domain-specific taxonomies can be integrated in a plug-and-play fashion.** For example, the domain taxonomy used for academic paper retrieval (Kang et al., 2024b) could be adopted as an alternative backbone in that context. Furthermore, if a well-defined taxonomy does not yet exist for a specific domain, one can be constructed using taxonomy induction methods (Zhang et al., 2018; Lee et al., 2022).

[5]https://pypi.org/project/selenium/

Dataset	# Docs	# Train queries	# Test queries
NQ320K	109,739	307,373	7,830
MS MARCO	323,569	366,235	5,187

Table 8: Statistics of the document retrieval datasets used.

A.4 Dataset Overview

In this work, we use NQ320K and MS MARCO. For NQ320K, we follow NCI (Wang et al., 2022) setup and adhered to the seen and unseen test splits used in GENRET (Sun et al., 2023). For MS MARCO, we construct dataset based on the MSMARCO document ranking dataset, following setups from Ultron (Zhou et al., 2022), GENRET (Sun et al., 2023), and NOVO (Wang et al., 2023). Table 8 shows the statistical details of the datasets used in our experiments.

A.5 Human Evaluation

We assess the quality of the generated explanations by conducting a human evaluation, where we compare the outputs of HyPE to other baseline models using Amazon Mechanical Turk (AMT). In this experiment, we use the title docid baseline described in Section 4.1, and additionally include BM25 as a baseline. which is capable of providing explanations for its retrieval results by highlighting the top-ranked terms contributing to the retrieval. We ask human judges to evaluate each sample's explanations based on the following four criteria.

- **Overall**: Which retrieval system output better explains the retrieval process overall?
- **Specificity**: Which retrieval system output provides more specific information?
- **Reasonability**: Which retrieval system output represents the retrieval process more logically and reasonably?

- **Comprehensiveness**: Which retrieval system output more comprehensively reflects the content of the document?

Note that our human evaluation involved a total of 300 human judges, with each sample being independently evaluated by 3 different human judges. This setting is designed by referencing previous works that conduct human evaluation (Kim et al., 2023; Lee et al., 2025). We show the interface for the human evaluation in Figure 5

A.6 Human Reranking

To evaluate whether explanations provided by HYPE can help users more effectively identify relevant documents in realistic search scenarios, we conduct a human reranking experiment via Amazon Mechanical Turk (AMT). We prepare two conditions for comparison: (1) a title-only setting and (2) a title+path setting, where the title is shown along with a hierarchical category path explanation generated by HYPE. For each query, five candidate documents are shown in both conditions, with the same title across settings; only the presence or absence of the category path differs, allowing for a controlled comparison of explanation impact. We randomly sample 100 query-document instances from the NQ320K dev set where the title docid baseline with HYPE successfully retrieves the gold document within the top-5 results. Human judges are asked to (1) rank the five candidates based on their relevance to the query (i.e., human reranking), and (2) indicate their confidence in the ranked list using a 5-point Likert scale. Based on the collected responses, we compute three metrics: Recall@1, which indicates whether the gold document was ranked first; MRR@5, which reflects how highly the gold document was ranked; and *Confidence*, which measures how certain participants are in their rankings. This setup allows us to quantitatively assess whether the explanations produced by HYPE improve both the accuracy and certainty of user decisions in realistic, information-limited search environments. We show the interface for the human reranking in Figure 6

A.7 Implementation Details

We use T5-base (Raffel et al., 2020) as our backbone model. For the input of the indexing task, we utilize the FirstP approach as our document representations (Section 2). Additionally, for the indexing task, we employ five synthetic queries, generated by using docT5query (Nogueira and Lin, 2020) with nucleus sampling with parameters $p = 0.8$ and $t = 0.8$. We use new [DOC] token to separate the path from the docid, which we insert between the path and the docid. We optimize our model as described in 3.2, while employing AdamW optimizer with a learning rate of 5e-4 and a batch size of 128, for up to 1M training steps. During the inference of HYPE, we adopt *path-aware ranking* strategy; for the path generation stage, we generate three category paths (i.e., $K_p = 3$), and for the docid generation stage, we use constrained beam search with a beam size of 100 (i.e., $m = 100$). To build the summary docid baseline and keyword docid baseline, we utilize the off-the-shelf text summarization model based on BART (Lewis et al., 2020) and the keyword extraction tool (Grootendorst, 2020).

A.8 Analysis of Efficiency

To quantify the inference cost introduced by generating hierarchical category paths, we measure the average inference time per instance using an NVIDIA RTX 4090 GPU. Specifically, we compare two decoding settings: (1) decoding only the docid, and (2) decoding both the docid and a single hierarchical category path. Our results show that the additional decoding required for generating a single path introduces only a marginal increase in inference time, demonstrating that HYPE 's explainability can be achieved with minimal efficiency loss.

A.9 Prompt

Table 9 shows the prompt used to construct the path candidate set for the document with LLM.

Prompt: Select candidate path set for document

You're a taxonomy expert. You will receive a document along with a set of candidate taxonomy hierarchy paths for the document. Your task is to select the path that can represent the document. Exclude paths that are too broad or less relevant or contain too specific information such as year.
You may list up to 3 paths, using only the paths in the candidate set. Do not include any explanation.

<Document title>: {Document title}
<Document contents>: {Document contents}
<Candidate hierarchy paths>: {pre-candidate path set}
<Selected hierarchy paths>: {Candidate path set}

Table 9: The prompt for building final candidate path set.

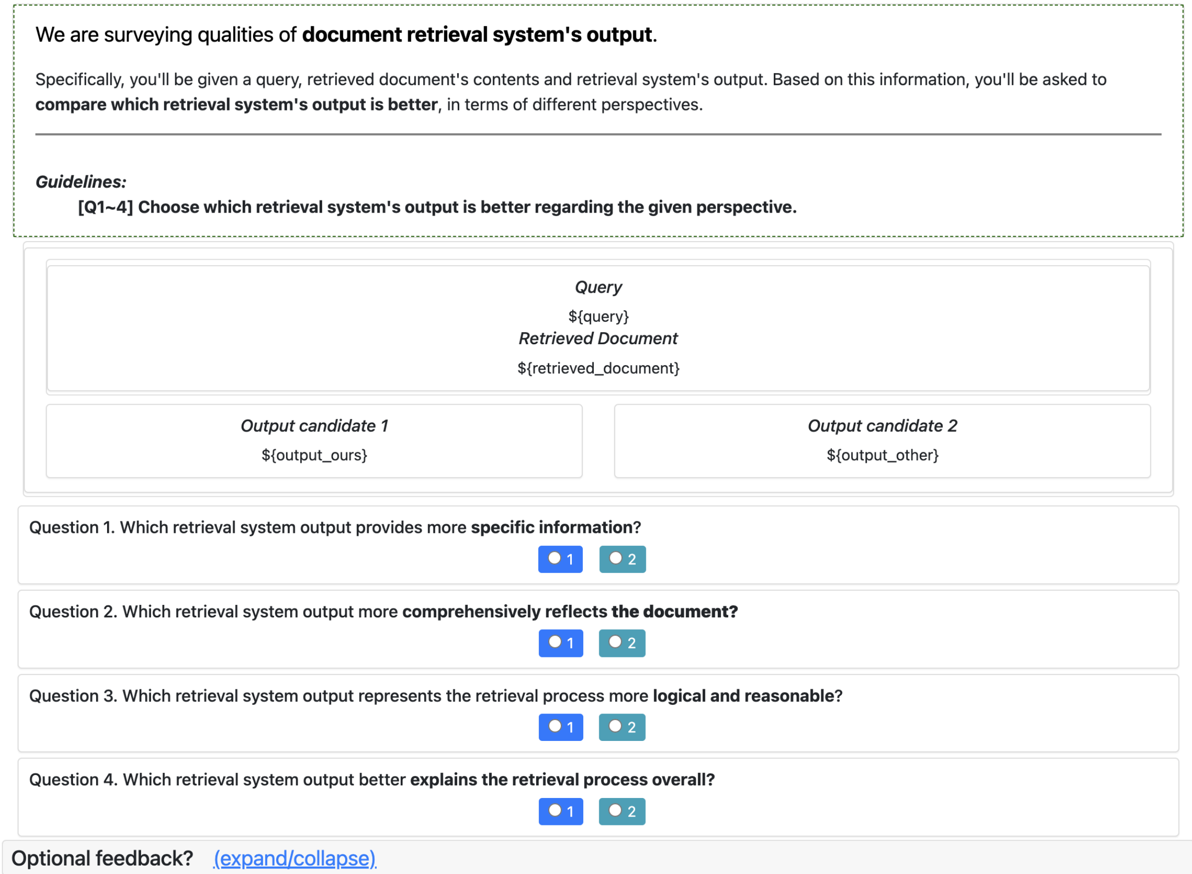
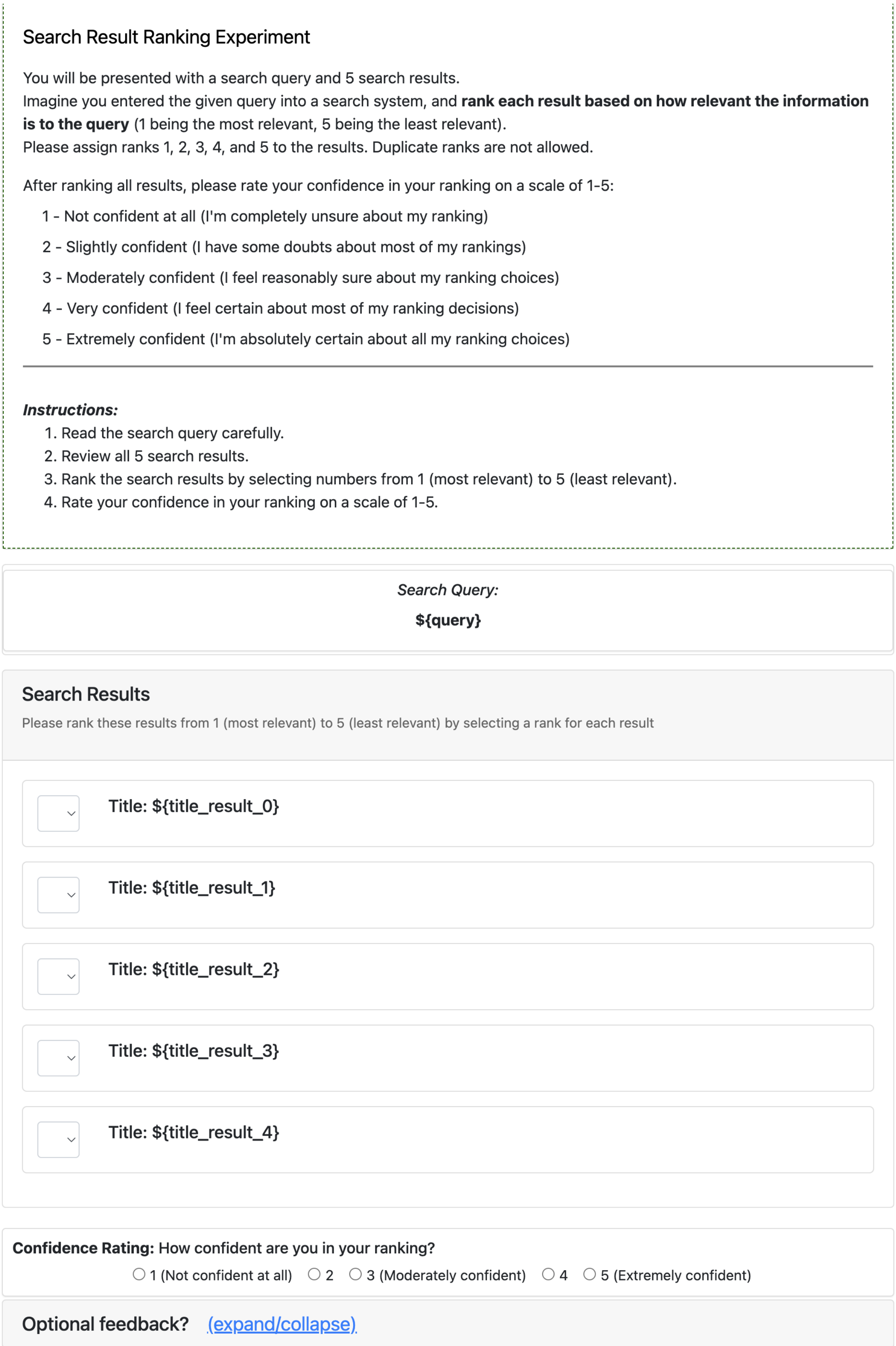


Figure 5: Annotator interface of human evaluation on retrieval system output.


Search Result Ranking Experiment

You will be presented with a search query and 5 search results.
Imagine you entered the given query into a search system, and **rank each result based on how relevant the information is to the query** (1 being the most relevant, 5 being the least relevant).
Please assign ranks 1, 2, 3, 4, and 5 to the results. Duplicate ranks are not allowed.

After ranking all results, please rate your confidence in your ranking on a scale of 1-5:

1 - Not confident at all (I'm completely unsure about my ranking)
2 - Slightly confident (I have some doubts about most of my rankings)
3 - Moderately confident (I feel reasonably sure about my ranking choices)
4 - Very confident (I feel certain about most of my ranking decisions)
5 - Extremely confident (I'm absolutely certain about all my ranking choices)

Instructions:

1. Read the search query carefully.
2. Review all 5 search results.
3. Rank the search results by selecting numbers from 1 (most relevant) to 5 (least relevant).
4. Rate your confidence in your ranking on a scale of 1-5.

Search Query:

${query}

Search Results

Please rank these results from 1 (most relevant) to 5 (least relevant) by selecting a rank for each result

Title: ${title_result_0}

Title: ${title_result_1}

Title: ${title_result_2}

Title: ${title_result_3}

Title: ${title_result_4}

Confidence Rating: How confident are you in your ranking?

○ 1 (Not confident at all) ○ 2 ○ 3 (Moderately confident) ○ 4 ○ 5 (Extremely confident)

Optional feedback? (expand/collapse)

Figure 6: Annotator interface of human reranking on retrieval system output.