# PaperRegister: Boosting Flexible-grained Paper Search via Hierarchical Register Indexing

**Zhuoqun Li**[1,2], **Xuanang Chen**[1], **Hongyu Lin**[1*], **Yaojie Lu**[1],
**Xianpei Han**[1], **Shanshan Jiang**[3], **Bin Dong**[3], **Le Sun**[1]
[1]Chinese Information Processing Laboratory, Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Ricoh Software Research Center Beijing, Ricoh Company, Ltd.
`{lizhuoqun2021,chenxuanang,hongyu,luyaojie,xianpei,sunle}@iscas.ac.cn`
`{shanshan.jiang,bin.dong}@cn.ricoh.com`

## Abstract

As researchers delve more deeply into their work, paper search requirements may become more flexible, sometimes involving specific details such as module configuration rather than being limited to coarse-grained topics. However, previous paper search systems are unable to meet these flexible-grained requirements, as previous systems mainly collect paper abstract to construct corpus index, which lacks detailed information to support retrieval by some finer-grained queries. In this work, we propose PaperRegister, which transforms traditional abstract-based index into a hierarchical index tree, thereby supporting queries at flexible granularity. Experiments on paper search tasks across a range of granularity demonstrate that PaperRegister achieves the SOTA performance, and particularly excels in the fine-grained scenarios, highlighting good potential as an effective solution for flexible-grained paper search in real-world applications. `https://github.com/icip-cas/PaperRegister`.

## 1 Introduction

Paper search is an important and almost everyday activity for researchers (Kuhlthau, 1991; Ellis et al., 1993; Hemminger et al., 2007; Case and Given, 2016). Typically, this process begins when user submits a natural-language query describing a topic, and then retrieval system matches this query against paper corpus and returns subset of papers with the highest relevance (Wadden et al., 2020; Cohan et al., 2020; Ajith et al., 2024; He et al., 2025). As researchers delve more deeply into their work, paper search requirements can become increasingly flexible. For example, the view of query may refer to detailed module configuration or methodological operation, rather than being limited to the level of coarse-grained topic (Mysore et al., 2021; Kang et al., 2024; Wang et al., 2023; Zhang et al., 2025). Therefore, a paper search system supporting queries at flexible granularity is of important value.

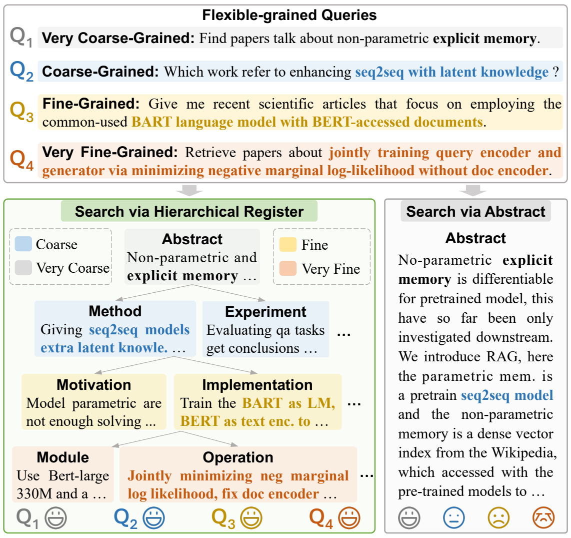


Figure 1: PaperRegister supports flexible-grained paper search via hierarchical register, while traditional method fails due to abstract cannot contain required details.

Unfortunately, existing paper search systems cannot effectively handle queries across flexible granularity, since they mainly use paper abstract to construct corpus index for retrieval (Zheng et al., 2020; Gao et al., 2023; Mackie et al., 2023; Lei et al., 2024). When the view of query involves finer-grained information that does not appear in abstract, they fail to retrieve relevant papers. As shown in Figure 1, the query *"Retrieve papers about jointly training encoder and generator via minimizing negative marginal log-likelihood without doc encoder"* focuses on a detailed training operation. For this query, the target papers cannot be successfully retrieved by traditional abstract-based index, because paper abstract does not mention such a detailed training operation. Therefore, building a paper search system supporting queries at flexible granularity remains a valuable and unresolved challenge.

To this end, we propose **PaperRegister**, which

---

[*] Corresponding author.

*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17561–17575
July 2-7, 2026

transforms traditional abstract-based index into a hierarchical index tree, thereby supporting queries at flexible granularity. Specially, PaperRegister includes offline hierarchical indexing and online adaptive retrieval. In the offline stage, to construct the hierarchical index tree for paper corpus, PaperRegister uses a hierarchical register schema, which is consisted of information nodes at various granularity, and each node path represents a kind of view. Based on this schema, PaperRegister first extracts fine-grained node contents, then aggregates node contents bottom-up, layer by layer, yielding hierarchical register for each paper. And then the hierarchical index tree is constructed by merging register of all papers. In the online stage, to adaptively retrieve via the hierarchical index tree, PaperRegister first employs a view recognizer to identify the views of query, which associates to suitable indexes from the tree, and then matches query against these indexes to achieve the precise paper search.

One key aspect of PaperRegister is the view recognizer, which must be both low-latency and high-accuracy. To this end, we construct the view recognizer by training a small-scale language model via hierarchical-reward reinforcement learning. Firstly, to alleviate online latency, we adopt a language model with 0.6 billion parameters as base. Then, to equip the model with basic identifying ability, we perform supervised fine-tuning, where query is the input and golden view is the label. Furthermore, to enhance accuracy of view recognizer, considering the hierarchical dependency of nodes in register schema, we design a hierarchical reward as reinforcement learning signal, which is calculated by the closeness level between predicted view and golden view in hierarchical register schema. And then we employ this reward on group relative policy optimization (GRPO) (Shao et al., 2024) to enhance the capability of view recognizer.

In experiments, we compare PaperRegister with several common-used methods on paper search, across various levels of granularity. Results show PaperRegister achieves the SOTA performance, with the improvement being more pronounced as query granularity is finer, confirming that PaperRegister is a robust solution for the challenging flexible-grained paper search task. The main contributions of this work can be summarized as follows:

- We construct PaperRegister, which can support paper search queries at flexible granularity via the hierarchical register indexing.
- We design hierarchical-reward reinforcement learning, which can train a powerful and low-latency view recognizer for PaperRegister.
- We conduct extensive experiments and analysis, which show PaperRegister is an advanced system for flexible-grained paper search.

## 2 PaperRegister

Existing paper search systems fail to handle queries at flexible granularity due to they primarily collect paper abstract to construct index of corpus, which does not contain detailed information to support queries at finer granularity. To this end, we propose PaperRegister, which can support flexible-grained paper search via hierarchical register indexing. As illustrated in Figure 2, in the offline stage, a hierarchical index tree is built for paper corpus, and in the online stage, adaptive retrieval is performed by selecting and using suitable indexes from the tree.

### 2.1 Task Formulation

The task involved in this work provides a query $q$ as input, with the goal of retrieving $M$ relevant papers $\{p^{(m)}\}_{m=1}^{M}$ based on the index $\mathcal{I}$ of paper corpus $\mathcal{C}$, which can be expressed as follows:

$$\{p^{(m)}\}_{m=1}^{M} = \mathcal{F}(q, \mathcal{I}) \tag{1}$$

For regular paper search, view of query $q$ is typically a coarse-grained topic that user is interested in, and systems mainly construct index $\mathcal{I}$ based on paper abstract to match against $q$. As user's requirements become more flexible, the view may refer to details such as module configuration and training operation. In this context, since this detailed information typically do not appear in abstract, existing paper search systems cannot retrieve accurately.

### 2.2 Offline Hierarchical Indexing

Considering the granularity of query view can be flexible across very fine-grained, moderately fine-grained, to coarse-grained, PaperRegister offline constructs a hierarchical index tree for paper corpus to support flexible paper search. Specifically, based on a hierarchical register schema, PaperRegister obtains hierarchical register for each paper through fine-grained content extracting and bottom-up content aggregating, and then hierarchical index tree is constructed by merging register of all papers.

**Hierarchical Register Schema.** To obtain the hierarchical register for each paper, we first design a hierarchical register schema, which consists of

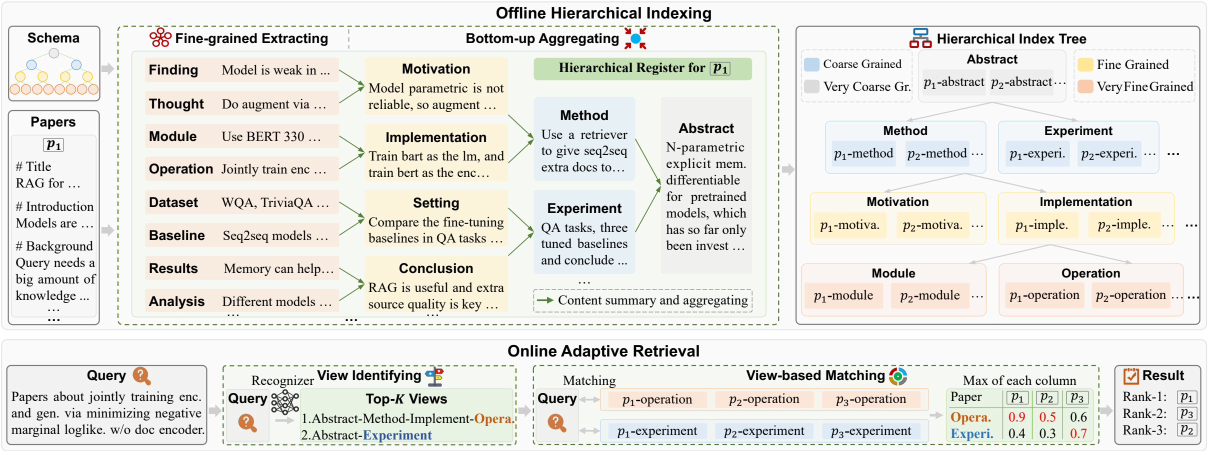


Figure 2: PaperRegister includes hierarchical indexing and adaptive retrieval. Offline, PaperRegister constructs hierarchical index tree via fine-grained content extracting and bottom-up content aggregating based on a hierarchical register schema. Online, PaperRegister first identify views of query and then conduct view-based matching.

information nodes as the following formula:

$$N_{i,j} = \{n_{i,j} : (c_{i,j},\ \{N_{i+1,j'}\}_{j'=1}^{Z})\} \qquad (2)$$

where $N_{i,j}$ is the $j$-th information node at the $i$-th layer, $n_{i,j}$ is node name, $c_{i,j}$ is expected node content from specific paper, $N_{i+1,j'}$ is a sub-node of $N_{i,j}$, $Z$ is number of sub-nodes under each node, and $i \in [1, ..., L]$, $L$ is number of layers in schema.

In hierarchical register schema, shallow-layer nodes represent coarse-grained information, and deep-layer nodes represent finer-grained information. For example, in the $i+1$-th layer, $N_{i+1,j'}$ and $N_{i+1,j'+1}$ respectively denote *module* configuration and training *operation* of paper, which are two kinds of relatively fine-grained information. While their common parent in the $i$-th layer, $N_{i,j}$, denotes method *implementation*, which is a coarser-grained summary of $N_{i+1,j'}$ and $N_{i+1,j'+1}$, thereby forming schema with hierarchical-dependency nodes.

In addition, since different types of papers have different styles and formats, we design five kinds of schema, corresponding to five paper types, and use a large language model to determine the type of each paper. Details are presented in Appendix A.

**Fine-grained Content Extracting.** Based on the above hierarchical register schema, in order to obtain fine-grained contents as much as possible, PaperRegister first extracts content for each information node at the $L$-th layer from paper $p^{(m)}$, which can be represented as the following formula:

$$c_{L,j}^{(m)} = \mathcal{M}_{extract}(p^{(m)}, n_{L,j}) \qquad (3)$$

where $n_{L,j}$ represents name of information node $N_{L,j}$ and $\mathcal{M}_{extract}$ is the extracting module.

To achieve accurate extraction, learning from several widely recognized works, in which large language models are proven to possess reliable capabilities for content extraction tasks (Edge et al., 2024; Li et al., 2025c), PaperRegister employs a large language model as extracting module. Specifically, PaperRegister takes the node name and text-formatted paper as input, uses instructions to guide in outputting corresponding content for the information node, and leaves it blank if the paper does not include corresponding content. The detailed process and instructions are in the Appendix B.

**Bottom-up Content Aggregating.** After extracting all the finest-grained contents, in order to smoothly obtain a complete hierarchical register, PaperRegister aggregates node contents layer by layer from bottom to top based on the hierarchical register schema, represented as following formula:

$$c_{i,j}^{(m)} = \mathcal{M}_{aggregate}(\{c_{i+1,j'}^{(m)}\}_{j'=1}^{Z}) \qquad (4)$$

where the layer $i$ is from $L-1$ to 1, thereby obtaining content for each information node in the hierarchical register schema for the paper $p^{(m)}$.

To achieve accurate aggregation, like extraction process, PaperRegister uses a large language model as the aggregating module $\mathcal{M}_{aggregate}$. Specifically, PaperRegister takes contents of sub-nodes as input and uses instructions to guide large language model in summarizing, condensing, and removing details to turn into summary text, thus obtaining content for upper-layer information node. The detailed process and instructions are in Appendix C.

At this point, PaperRegister obtains node contents at various granular levels to compose the hier-

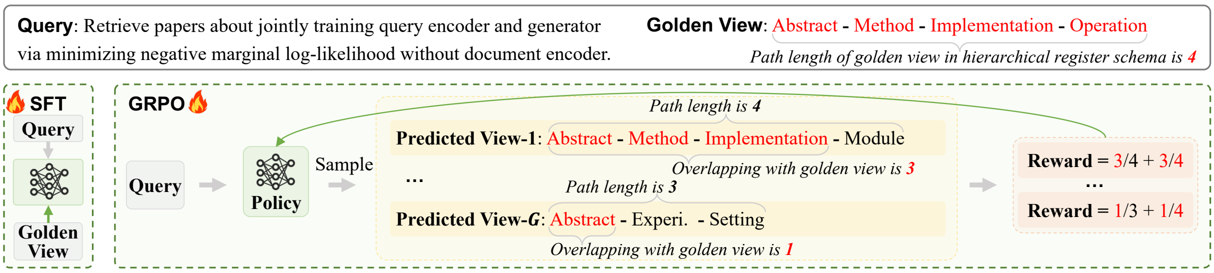


Figure 3: Illustration of view recognizer training, including SFT and GRPO via hierarchical reward, which is calculated based on the closeness level of predicted view and golden view in the hierarchical register schema.

archical register for $p^{(m)}$. And then merges register of all papers in corpus $\mathcal{C}$ to construct hierarchical index tree $\mathcal{I}_h$, as shown in following formula:

$$\mathcal{I}_h = \{\{\mathrm{I}_{i,j}\}_{j=1}^{Z}\}_{i=1}^{L} = \{\{\mathcal{M}_{idx}\{c_{i,j}^{(m)}\}_{m=1}^{|\mathcal{C}|}\}_{j=1}^{Z}\}_{i=1}^{L} \qquad (5)$$

where $\mathcal{M}_{idx}$ is the indexing module such as BM25 and DPR, each $\mathrm{I}_{i,j}$ is an index in hierarchical index tree, corresponding to a kind of view for corpus.

### 2.3 Online Adaptive Retrieval

Based on the offline hierarchical indexing above, PaperRegister perform online adaptive retrieval via suitable indexes from hierarchical index tree. Specifically, PaperRegister first identifies views involved in the query, determining corresponding indexes from hierarchical index tree, then conduct retrieval by matching query against these indexes.

**View Identifying.** To achieve adaptive retrieval, PaperRegister first uses a view recognizer to identify the view $v_k$ in query $q$, represented as follows:

$$\{v_k\}_{k=1}^{K} = \mathcal{M}_{identify}(q) \qquad (6)$$

where the candidate set of $v_k$ is all node paths in the hierarchical register schema, $\mathcal{M}_{identify}$ is employed base on a small-scale language model with special training, which will be explained in next section. And to ensure that identifying results can cover the real views of query as more as possible, $\mathcal{M}_{identify}$ uses the beam search strategy (Holtzman et al., 2020) to sample the top-$K$ output views.

**View-based Matching.** After identifying views in query, PaperRegister looks up corresponding indexes from hierarchical index tree $\mathcal{I}_h$, as follows:

$$\{\mathrm{I}_k\}_{k=1}^{K} = \mathcal{M}_{lookup}(\mathcal{I}_h, \{v_k\}_{k=1}^{K}) \qquad (7)$$

where $\mathrm{I}_k$ is consisted of $\{c_k^{(m)}\}_{m=1}^{|\mathcal{C}|}$ as Formula 5.

Then these indexes are used to calculate relevance score of input query $q$ and each paper $p^{(m)}$ :

$$s(q, p^{(m)}) = \max\{\mathcal{M}_{rel}(q,\ c_k^{(m)})_{k=1}^{K}\} \qquad (8)$$

where $\mathcal{M}_{rel}$ is relevance module such as BM25.

Finally, papers with top-$M$ relevance score, $\{p^{(m)}\}_{m=1}^{M}$, are selected as final results for query.

## 3 View Recognizer Training

For PaperRegister system, view recognizer is a key module, which must both alleviate latency and ensure accuracy. To this end, as in Figure 3, we use a small-scale language model with 0.6B parameters as base, first do supervised fine-tuning (SFT), and then further enhance capability by hierarchical-reward group relative policy optimization (GRPO).

**Training Data.** The format of training dataset to support SFT and GRPO is shown as follows:

$$\mathcal{D}_{train} = \{q_j, v_j\}_{j=1}^{|\mathcal{D}_{train}|} \qquad (9)$$

where $q_j$ is a query, $v_j$ is golden view of query, which is node path in hierarchical register schema.

**Supervised Fine-tuning.** To give small-scale view recognizer basic identifying ability and make it easier for subsequent reinforcement learning, we first perform SFT by minimizing following loss:

$$\mathcal{L}_{SFT}(\theta) = -\mathbb{E}_{(q,v)\sim\mathcal{D}_{train}} \sum_{t=1}^{|v|} \log \pi_\theta(v_t|q, v_{<t}) \qquad (10)$$

where $\pi_\theta$ is model, $v_t$ is the $t$-th golden view token.

**Hierarchical-reward GRPO.** To strengthen the capability of small-scale view recognizer, we conduct GRPO (Shao et al., 2024) with hierarchical reward, which is calculated via the closeness level of predicted view and golden view in hierarchical register schema. Specially, considering the hierarchical-dependency feature of register schema, different wrong predicted views should not be treated equally. For example, the golden view of query is represented as *Abstract-Method-Implementation-Operation* in schema, and two predicted views are *Abstract-Method-Implementation-Module* and *Abstract-Experiment-Dataset* respectively. Although both are wrong, the first one is better than the second because it is closer to golden view in hierarchical register schema, suggesting a higher information overlap with golden view.

Based on this, we measure path overlap between predicted view $\hat{v}_j$ and golden view $v_j$ in hierarchical register schema, obtaining hierarchical reward:

$$r = \frac{\mathcal{M}_{overlap}(v_j, \hat{v}_j)}{|\hat{v}_j|} + \frac{\mathcal{M}_{overlap}(v_j, \hat{v}_j)}{|v_j|} \quad (11)$$

where $\mathcal{M}_{overlap}$ returns overlap level of inputs, $|\hat{v}_j|$ means path length of predicted view in hierarchical register schema, and $|v_j|$ is that of golden view.

Then this hierarchical reward is used in GRPO, which is by maximizing the following objective:

$$\begin{aligned} \mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{[q \sim D_{train}, \{v_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(v|q)]} \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|v_i|} \sum_{t=1}^{|v_i|} \Bigg\{ \\ \min \left[ \frac{\pi_\theta (v_{i,t} \mid q, v_{i,<t})}{\pi_{\theta_{\text{old}}} (v_{i,t} \mid q, v_{i,<t})} \hat{A}_{i,t}, \text{clip}\left( \frac{\pi_\theta (v_{i,t} \mid q, v_{i,<t})}{\pi_{\theta_{\text{old}}} (v_{i,t} \mid q, v_{i,<t})}, 1-\epsilon, 1+\epsilon \right) \hat{A}_{i,t} \right] \\ - \beta \mathbb{D}_{\text{KL}} [\pi_\theta \parallel \pi_{\text{ref}}] \Bigg\}, \text{ where } \hat{A}_{i,t} = \frac{r_i - mean(\{r_1, r_2, ..., r_G\})}{std(\{r_1, r_2, ..., r_G\})} \end{aligned} \quad (12)$$

where $\pi_{\theta_{\text{old}}}$ is initialized by SFT model, $G$ is number of each group, $\epsilon$ and $\beta$ are hyper-parameters.

# 4 Experiments

## 4.1 Experimental Settings

**Datasets.** To conduct experiments for flexible-grained paper search, we first use *LitSearch* (Ajith et al., 2024), which mainly contains coarse-grained queries, and then we build *Flexible-grained Search*, a new dataset covering queries across various granularity, including data for test, development and training. The metrics reported in experiments are **R@5** (recall@5) and **R@10** (recall@10). And detailed statistical information is in the Appendix D.

(1) **LitSearch** (Ajith et al., 2024). By using GPT-4 to rewrite citations in papers and asking authors to write, this dataset constructs 597 paper search queries, mainly involving coarse-grained topics like *"Where can I find some researches on evaluating consistency in generated summaries"*.

(2) **Flexible-grained Search**. We collect 4,200 papers in the arXiv platform as corpus and perform the offline hierarchical indexing. To build training and development data, we pick 2,100 paper registers, and generate query based on each node content in each register by LLM, where the node path is golden view of query. Then we randomly split, getting 13,824 training data and 695 development data, training data are used in view recognizer training, and development data are for analysis in Table 3. The format of training and development data is $\{q_j, v_j, p_j\}$. To build test data across various granularity, we pick the other 2,100 paper registers. And in order to prevent data leakage, we do not directly use node content to generate query. Instead, we employ LLM to find original paper text related to each node, then generate queries based on these text, obtaining 5,644 test data. Thanks to the hierarchical feature of register, test data cover: **F.g.Search-1** (general granularity), **F.g.Search-2** (fine granularity), and **F.g.Search-3** (very fine granularity). Format of each test data is $\{q_j, p_j\}$, and LLM in this section is Qwen3-32B (Qwen, 2025).

**Selected Baselines.** We select baselines from commonly used or advanced methods suitable for paper search tasks. (1) **Direct Matching.** These methods directly use the paper title or abstract to construct corpus index, and then perform retrieval by matching query with these contents. In addition, to demonstrate that the effectiveness of PaperRegister does not simply stem from using total paper text, we include a method that constructs the corpus index using the total paper text into this category of baselines. (2) **Query Enhancing.** These methods aim to improve paper search performance by paraphrasing the input query, including: Rewriting (Ma et al., 2023), which uses a LLM to rewrite original query. HyDE (Gao et al., 2023), which uses a LLM to generate a fake document based on input query and retrieves real documents by this fake document. CSQE (Lei et al., 2024), which initially retrieves several documents and then uses a LLM to expand original query based on these initially-retrieved documents. We use Qwen3-32B as LLM for these baselines. (3) **Multi-field Indexing.** These methods split original document into multiple parts, get flat multi-field index, then calculate similarity by selected indexes and integrate to determine overall relevance (Li et al., 2025b). In experiments, due to the lack of a universal multiple-filed partitioning for all kinds of paper, we employ four settings, including splitting the original paper to fixed length of 512-token chunk or by raw paragraph, taking average similarity or maximum similarity of all parts as final score, represented as Chunk$_{avg}$, Chunk$_{max}$, Paragraph$_{avg}$, and Paragraph$_{max}$, respectively.

**Implementation Details.** In the offline stage, we employ Qwen3-32B (Qwen, 2025) as the large language model in process of fine-grained content extracting and bottom-up content aggregating, and deploy it as API by vllm[1] on two A-100 80G GPU for convenience. In the online stage, we set $K$ as 5

---

[1] https://pypi.org/project/vllm/

| **Method** | **BM25-based Paper Search** | | | | | | | | **DPR-based Paper Search** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **LitSearch** | | **F.g.Search-1** | | **F.g.Search-2** | | **F.g.Search-3** | | **LitSearch** | | **F.g.Search-1** | | **F.g.Search-2** | | **F.g.Search-3** | |
| | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 |
| Direct Matching | | | | | | | | | | | | | | | | |
| Title | 54.2 | 59.5 | 42.0 | 47.3 | 36.1 | 42.0 | 30.4 | 35.3 | 66.8 | 73.1 | 52.0 | 58.8 | 45.8 | 52.1 | 40.6 | 47.1 |
| Abstract | 67.1 | 71.3 | 63.7 | 69.0 | 57.4 | 62.9 | 54.2 | 59.7 | 77.7 | 83.2 | 69.4 | 74.3 | 62.1 | 68.1 | 58.2 | 63.1 |
| Total Paper | 64.2 | 68.9 | 79.7 | 82.4 | 81.4 | 83.5 | 84.2 | 86.2 | 74.8 | 81.9 | 72.9 | 78.7 | 68.8 | 73.8 | 65.8 | 71.0 |
| Query Enhancing | | | | | | | | | | | | | | | | |
| Rewriting (Ma et al., 2023) | 61.9 | 69.6 | 58.4 | 64.2 | 54.1 | 59.8 | 50.9 | 56.3 | 76.0 | 83.2 | 67.1 | 72.6 | 60.4 | 65.2 | 55.2 | 60.9 |
| HyDE (Gao et al., 2023) | 68.9 | 75.4 | 60.3 | 66.3 | 54.3 | 60.0 | 51.5 | 57.6 | 76.5 | 83.4 | 66.8 | 72.3 | 58.9 | 65.7 | 56.3 | 62.4 |
| CSQE (Lei et al., 2024) | 69.0 | 73.8 | 59.1 | 64.4 | 54.2 | 58.6 | 51.3 | 55.9 | 77.9 | 81.7 | 65.8 | 71.1 | 60.2 | 65.5 | 55.2 | 60.9 |
| Multi-field Indexing | | | | | | | | | | | | | | | | |
| Chunk$_{avg}$ (Li et al., 2025b) | 58.1 | 67.6 | 68.1 | 74.2 | 66.6 | 71.9 | 65.9 | 71.6 | 49.7 | 58.6 | 51.1 | 58.3 | 46.4 | 53.4 | 46.4 | 52.2 |
| Chunk$_{max}$ (Li et al., 2025b) | 67.9 | 75.5 | 80.0 | 83.1 | 81.4 | 84.7 | 85.3 | 87.6 | 72.6 | 79.7 | 79.8 | 83.1 | 76.6 | 80.5 | 79.0 | 82.1 |
| Paragraph$_{avg}$ (Li et al., 2025b) | 29.7 | 38.3 | 58.9 | 66.6 | 59.8 | 66.5 | 58.2 | 66.2 | 20.5 | 22.8 | 23.3 | 30.6 | 23.1 | 29.8 | 22.4 | 29.2 |
| Paragraph$_{max}$ (Li et al., 2025b) | 64.3 | 70.9 | 73.8 | 78.8 | 75.4 | 79.9 | 80.8 | 83.9 | 79.5 | 85.0 | 79.2 | 82.8 | 76.5 | 81.4 | 78.8 | 81.8 |
| Hierarchical Register Indexing | | | | | | | | | | | | | | | | |
| PaperRegister (Ours) | **69.7** | **76.4** | **89.7***  | **90.9*** | **88.0*** | **89.0*** | **87.5*** | **88.7*** | **81.0*** | **87.1*** | **84.1*** | **87.1*** | **79.9*** | **82.5*** | **80.8*** | **82.9*** |

Table 1: Main results on paper search across various granularity, where granularity is coarse-to-fine from LitSearch to F.g.Search-3. Results shows that PaperRegister is an effective system for flexible-grained paper search and the advantage of PaperRegister become more pronounced at finer granular tasks. The * indicates statistical significance ($p < 0.05$) compared with the best baseline. And experiment results by more metrics are reported in Appendix E.

and $M$ as 5 or 10, use Qwen3-0.6B (Qwen, 2025) as the base model of view recognizer, employ a prefix tree-based restricted decoding strategy (Tang et al., 2024) in view identifying process to prevent the model outputting irrelevant token, and conduct view-based matching process via rank-bm25[2] for BM25 and gte-Qwen2-7B-instruct (Li et al., 2023) for DPR. For the view recognizer training, we use TRL[3] framework to conduct SFT and GRPO, with train epoch as 5 in SFT, $G$ as 5 and train epoch as 2 in GRPO, and all other parameters as default value.

## 4.2 Overall Results

Results compared with baselines are shown in the Table 1, there are two main conclusions:

(1) **PaperRegister is an effective system to addressing flexible-grained paper search.** As shown in the Table 1, PaperRegister demonstrates excellent performance in paper search tasks at various granularity, outperforming all baselines in both BM25-based matching and DPR-based matching settings, on both recall@5 and recall@10 metrics. For example, in F.g.Search-3, under the DPR-based matching setting, PaperRegister achieves a recall@5 score of 80.8, while using abstract-based index yields only 58.2, where PaperRegister can achieve a performance improvement of 22.6. All in all, PaperRegister can improve a lot compared with various previous paper search methods.

(2) **Compared to traditional methods, advantage of PaperRegister become more pronounced at finer granular queries.** According to experimental setting, from LitSearch to F.g.Search-1 to F.g.Search-2 to F.g.Search-3, the granularity of query becomes increasingly finer. Experimental results show performance improvement of PaperRegister over abstract-based method becomes larger. For example, under the DPR-based matching setting, PaperRegister achieves recall@5 scores of 81.0, 84.1, 79.9, and 80.8 on the four datasets, respectively, while using the abstract-based index yields scores of 77.7, 69.4, 62.1, and 58.2. Here, PaperRegister delivers performance improvements of 3.3, 14.7, 17.8, and 22.6, respectively. This can to some extent validate rationality of our motivation, traditional paper search methods via abstract-based index struggles to handle fine-grained queries.

## 4.3 Ablation Results

To validate importance of hierarchical register indexing, we employ ablation by simplifying register schema. As shown in Table 2, where "w/ only layer-1" means retaining only the coarsest-grained nodes, "w/ only layer-2" means retaining only the medium-grained nodes, and "w/ only layer-3" means retaining only the finest-grained nodes. Base on the ablation results, we get the following conclusions:

(1) **Hierarchical register indexing is important for ensuring accurate paper search.** Compared to using complete hierarchical register schema, using a schema composed of nodes from any single layer leads to obvious performance drop.

---

[2] https://pypi.org/project/rank-bm25/

[3] https://github.com/huggingface/trl

| Method | BM25-based Paper Search | | | | | | | | DPR-based Paper Search | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LitSearch | | F.g.Search-1 | | F.g.Search-2 | | F.g.Search-3 | | LitSearch | | F.g.Search-1 | | F.g.Search-2 | | F.g.Search-3 | |
| | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 | R@5 | R@10 |
| PaperRegister | **69.7** | **76.4** | **89.7** | **90.9** | **88.0** | **89.0** | **87.5** | **88.7** | **81.0** | **87.1** | **84.1** | **87.1** | **79.9** | **82.5** | **80.8** | **82.9** |
| w/ only layer-1 | 64.5 | 70.4 | 87.8 | 90.6 | 77.4 | 80.2 | 73.5 | 76.7 | 79.0 | 83.5 | 82.5 | 85.2 | 73.4 | 77.1 | 68.1 | 72.2 |
| w/ only layer-2 | 66.8 | 73.8 | 84.4 | 86.5 | 87.1 | 88.8 | 82.0 | 84.4 | 78.9 | 84.2 | 81.1 | 84.9 | 79.3 | 82.2 | 73.5 | 77.9 |
| w/ only layer-3 | 64.6 | 71.7 | 79.9 | 81.8 | 83.7 | 85.0 | 86.8 | 88.1 | 78.8 | 85.2 | 77.8 | 81.1 | 77.3 | 80.0 | 80.4 | 82.6 |

Table 2: Ablation for hierarchical register indexing. Results show that hierarchical index tree is important for flexible-grained paper search and different layers in hierarchical register schema serve queries at different granularity.

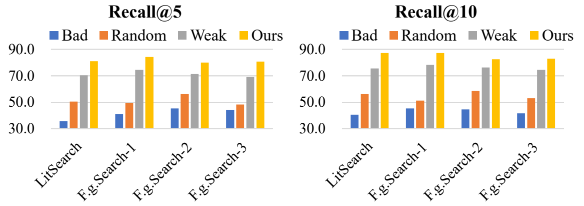


Figure 4: Performance of PaperRegister with different view recognizer. The figure shows a strong recognizer is with obvious positive impact on the overall system.

| Recognizer | ACC ↑ | Latency (s) ↓ |
|---|---|---|
| Qwen3-32B | 30.5 | 28.3 |
| Qwen3-32B (few-shot) | 47.8 | 37.8 |
| View Recognizer in PaperRegister | **83.5** | **2.3** |
| w/ only SFT | 80.9 | - |
| w/o hierarchical reward | 81.7 | - |

Table 3: Comparison and ablation for the view recognizer training. The table shows that the training process in this work is an effective approach to obtain a view recognizer with both high performance and low latency.

For example, recall@5 of LitSearch under BM25-based matching drops from 69.7 to 64.5, 66.8, 64.6 for "w/ only layer-1", "w/ only layer-2", "w/ only layer-3", respectively. And other datasets also show similar situation. Therefore, results strongly prove the importance of hierarchical register indexing.

(2) **Different layers in hierarchical register schema serve queries at different granularity.** For example, under "w/ only layer-1" setting, the DPR-based matching and recall@5 metric of F.g.Search-1 drops from 84.1 to 82.5 (1.6 decrease), while F.g.Search-1 is from 80.8 to 68.1 (12.7 decrease), which is significantly larger than 1.6. Conversely, under "w/ only layer-3" setting, F.g.Search-1 drops from 84.1 to 77.8 (6.3 decrease), while F.g.Search-3 is from 80.8 to 80.4 (0.4 decrease), which is much smaller than 6.3. Therefore, nodes of different layers play specific roles in handling queries of different granularity, further strongly proving necessity of hierarchical register indexing.

## 4.4 Detailed Analysis

In this section, we conduct detailed analysis to explore the compatibility and real-world practicality of PaperRegister. Due to the space constraint, we only report DPR-based performance for analysis.

**Effect of View Recognizer and Training.** In order to strictly analyze the role of view recognizer in PaperRegister and the necessity of our training, we first examine relationship between the capability of view recognizer and the final performance of PaperRegister in Figure 4, then compare with Qwen3-32B and conduct ablation in Table 3. The detailed process and findings are in following:

As shown in Figure 4, where "Bad Recognizer","Random Recognizer","Weak Recognizer" represent the completely incorrect, randomly predicting, and weak-performing view recognizers, respectively, the curve demonstrates a clear positive correlation between the capability of view recognizer and final performance of PaperRegister. *Therefore, building an accurate view recognizer is a key factor for the PaperRegister system to achieve high performance in complex paper search.*

As shown in Table 3, We compare accuracy and latency of view identifying between the view recognizer in PaperRegister and Qwen3-32B (enable thinking) under zero-shot and few-shot settings. The results show the view recognizer in PaperRegister is with better accuracy and latency than powerful Qwen3-32B. Furthermore, we only keep SFT for the view recognizer training, or replace the hierarchical reward with direct 0-1 reward in GRPO training. Results show that accuracy decreases to some extent. *Therefore, hierarchical-reward GRPO is effective to enhance the view recognizer achieving both high performance and low latency.*

**Compatibility with PaSa.** Given that some complex information retrieval frameworks such as PaSa (He et al., 2025) incorporate various modules like rewriting, retrieval, iteration, and filtering, to explore whether PaperRegister can be compatible

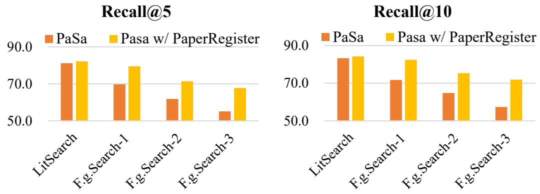


Figure 5: Performance of combining PaperRegister into PaSa framework. The figure shows that PaperRegister can greatly cooperate with complex modules in PaSa.

with such complex frameworks and further enhance paper search performance, we conduct experiment by replacing the original search module in PaSa with PaperRegister. As shown in Figure 5, PaperRegister can further improve performance of PaSa on paper search tasks across various granularity. *Therefore, PaperRegister can be effectively adapted as a search system into complex frameworks and further enhance the paper search capability.*

**Online Search Efficiency.** Considering that the efficiency of online paper search system is a very crucial aspect for ensuring its usability in real-world scenarios, we compare the online search efficiency of PaperRegister with multiple baselines. As shown in the Table 4, the online latency of these baselines is significantly higher than that of PaperRegister, which limits the practical applicability of these methods in real-world scenarios. *In contrast, PaperRegister demonstrates relatively acceptable online latency, proving it has a good potential for application in the real-world paper search tasks.*

| **Method** | **Avg. R@5** | **Avg. R@10** | **Online Latency** (s) ↓ |
|---|---|---|---|
| Rewriting | 64.7 | 70.5 | 9.3 |
| HyDE | 64.6 | 70.9 | 20.7 |
| CSQE | 64.8 | 69.8 | 33.5 |
| Chunk$_{max}$ | 77.0 | 81.4 | 5.4 |
| PaperRegister (Ours) | **81.5** | **84.9** | **2.5** |

Table 4: Comparison of online latency. PaperRegister is with less latency for better real-world applicability.

**Reducing Indexing Time.** Although PaperRegister achieves excellent performance, it requires a relatively long time for offline indexing. Therefore, we explore whether it can sacrifice a certain amount of performance to reduce time consumption. As shown in Table 5, we replace original two-step registering (extracting and aggregating) with directly extracting all contents. Results show this can significantly reduce the offline indexing time with acceptable performance degradation, which *provides an alternative approach allowing a trade-off based on specific requirements in real-world application.*

| **Method** | **Avg. R@5** | **Avg. R@10** | **Indexing Time** (h) ↓ |
|---|---|---|---|
| Abstract | 66.9 | 72.1 | / |
| PaperRegister (Ours) | **81.5** | **84.9** | 47.6 |
| w/o two-step registering | 76.1 | 78.9 | **14.5** |

Table 5: PaperRegister can support sacrificing certain performance to alleviate offline time consumption.

## 5 Related Work

**Query Enhancing.** Previous works mainly improve paper search by query enhancing, such as using LLM rewriting (Ma et al., 2023; Anand et al., 2023), generating pseudo-documents to replace original query (Gao et al., 2023; Li et al., 2024a), employing powerful models as agent system for multi-round expansion (He et al., 2025; Ren et al., 2025), augmenting via initially retrieved documents (Lei et al., 2024; Li et al., 2025a), or extracting keywords from corpus to enrich query (Kang et al., 2024; Zhang et al., 2025). Although these methods can improve regular paper search, they fail to address flexible-grained paper search because do not solve fundamental flaw of abstract-based index.

**Multi-filed Indexing.** Recently, several studies improve by splitting documents into multiple parts to build a flat, multi-field index (Sotaro et al., 2024; Li et al., 2025b; Shi et al., 2025; Chen et al., 2025). However, these works are mainly limited to a single granularity level, still struggling to adapt to paper search that demands varying degrees of granularity. In contrast, PaperRegister fundamentally differs by introducing a hierarchical register index. Tree-structured organization contains information across various granularity, enabling system to conduct matching at the appropriate levels, thereby effectively supporting flexible-grained paper search.

## 6 Conclusion

In this work, we propose PaperRegister, which can support paper search queries at flexible granularity. Specially, PaperRegister offline constructs a hierarchical index tree and online adaptively retrieve based on this tree. Furthermore, we design a powerful and low-latency view recognizer by applying supervised finetuning and hierarchical-reward group relative policy optimization. Experiments on flexible-grained paper search tasks demonstrate PaperRegister is an effective solution, with improvement being more pronounced as granularity finer. This work offers a promising direction for developing more powerful paper search systems in future.

## 7 Limitations

PaperRegister transforms raw paper into hierarchical paper register, thereby achieving strong performance in flexible-grained paper search. The main limitations are on the consumption associated with LLM utilization and risks stemming from the inherent limitations of LLM. Firstly, using LLM for offline hierarchical indexing requires substantial computational and storage resources. When the paper corpus is large, this may place high demands on computer hardware. Secondly, our experiments are conducted only on normal paper corpus, where LLM could effectively accomplish the specified task. We have not yet do validate under extreme conditions, such as with incomplete papers or papers from niche domains, primarily due to a lack of relevant experimental data. Therefore, mitigating computational resource consumption of PaperRegister, as well as verifying and improving effectiveness and robustness across a broader range of paper, remain important directions for the future work.

## Acknowledgments

We sincerely thank the reviewers for their insightful comments and valuable suggestions. This work was supported by the National Key R&D Program of China (2024YFC3308000), the Natural Science Foundation of China (No. 62572456, 62476265, 62506354).

## References

Anirudh Ajith, Mengzhou Xia, Alexis Chevalier, Tanya Goyal, Danqi Chen, and Tianyu Gao. 2024. LitSearch: A retrieval benchmark for scientific literature search. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15068–15083, Miami, Florida, USA. Association for Computational Linguistics.

Abhijit Anand, Vinay Setty, Avishek Anand, and 1 others. 2023. Context aware query rewriting for text rankers using llm. *arXiv preprint arXiv:2308.16753*.

Donald O Case and Lisa M Given. 2016. *Looking for information: A survey of research on information seeking, needs, and behavior*. Emerald Group Publishing.

Peter Baile Chen, Tomer Wolfson, Michael Cafarella, and Dan Roth. 2025. Enrichindex: Using llms to enrich retrieval indices offline. *arXiv preprint arXiv:2504.03598*.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online. Association for Computational Linguistics.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *ArXiv preprint*, abs/2404.16130.

David Ellis, Deborah Cox, and Katherine Hall. 1993. A comparison of the information seeking patterns of researchers in the physical and social sciences. *Journal of documentation*, 49(4):356–369.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Yichen He, Guanhua Huang, Peiyuan Feng, Yuan Lin, Yuchen Zhang, Hang Li, and Weinan E. 2025. PaSa: An LLM agent for comprehensive academic paper search. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11663–11679, Vienna, Austria. Association for Computational Linguistics.

Bradley M Hemminger, Dihui Lu, KTL Vaughan, and Stephanie J Adams. 2007. Information seeking behavior of academic scientists. *Journal of the American society for information science and technology*, 58(14):2205–2225.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

SeongKu Kang, Yunyi Zhang, Pengcheng Jiang, Dongha Lee, Jiawei Han, and Hwanjo Yu. 2024. Taxonomy-guided semantic indexing for academic paper search. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7169–7184, Miami, Florida, USA. Association for Computational Linguistics.

Carol C Kuhlthau. 1991. Inside the search process: Information seeking from the user's perspective. *Journal of the American society for information science*, 42(5):361–371.

Yibin Lei, Yu Cao, Tianyi Zhou, Tao Shen, and Andrew Yates. 2024. Corpus-steered query expansion with large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 393–401, St. Julian's, Malta. Association for Computational Linguistics.

Hang Li, Xiao Wang, Bevan Koopman, and Guido Zuccon. 2025a. Pseudo relevance feedback is enough to close the gap between small and large dense retrieval models. *arXiv preprint arXiv:2503.14887*.

Lei Li, Xiangxu Zhang, Xiao Zhou, and Zheng Liu. 2024a. Automir: Effective zero-shot medical information retrieval without relevance labels. *arXiv preprint arXiv:2410.20050*.

Millicent Li, Tongfei Chen, Benjamin Van Durme, and Patrick Xia. 2025b. Multi-field adaptive retrieval. In *The Thirteenth International Conference on Learning Representations*.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *ArXiv preprint*, abs/2308.03281.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2024b. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. *ArXiv preprint*, abs/2410.08815.

Zhuoqun Li, Hongyu Lin, Yaojie Lu, Hao Xiang, Xianpei Han, and Le Sun. 2024c. Meta-cognitive analysis: Evaluating declarative and procedural knowledge in datasets and large language models. *arXiv preprint arXiv:2403.09750*.

Zhuoqun Li, Haiyang Yu, Xuanang Chen, Hongyu Lin, Yaojie Lu, Fei Huang, Xianpei Han, Yongbin Li, and Le Sun. 2025c. DeepSolution: Boosting complex engineering solution design via tree-based exploration and bi-point thinking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4380–4396, Vienna, Austria. Association for Computational Linguistics.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.

Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. 2023. Generative relevance feedback with large language models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2026–2031. ACM.

Sheshera Mysore, Tim O'Gorman, Andrew McCallum, and Hamed Zamani. 2021. CSFCube - a test collection of computer science research articles for faceted query by example. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Team Qwen. 2025. Qwen3 technical report. *ArXiv preprint*, abs/2505.09388.

Shuo Ren, Pu Jian, Zhenjiang Ren, Chunlin Leng, Can Xie, and Jiajun Zhang. 2025. Towards scientific intelligence: A survey of llm-based scientific agents. *arXiv preprint arXiv:2503.24047*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, abs/2402.03300.

Jimeng Shi, Sizhe Zhou, Bowen Jin, Wei Hu, Shaowen Wang, Giri Narasimhan, and Jiawei Han. 2025. Hypercube-rag: Hypercube-based retrieval-augmented generation for in-domain scientific question-answering. *arXiv preprint arXiv:2505.19288*.

Takeshita Sotaro, Ponzetto Simone, Eckert Kai, and Takeshita Sotaro. 2024. GenGO: ACL paper explorer with semantic features. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 117–126, Bangkok, Thailand. Association for Computational Linguistics.

Qiaoyu Tang, Jiawei Chen, Zhuoqun Li, Bowen Yu, Yaojie Lu, Cheng Fu, Haiyang Yu, Hongyu Lin, Fei Huang, Ben He, Xianpei Han, Le Sun, and Yongbin Li. 2024. Self-retrieval: End-to-end information retrieval with one large language model. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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

Jianyou Wang, Kaicheng Wang, Xiaoyue Wang, Prudhviraj Naidu, Leon Bergen, and Ramamohan Paturi. 2023. Scientific document retrieval using multi-level aspect-based queries. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yunyi Zhang, Ruozhen Yang, Siqi Jiao, SeongKu Kang, and Jiawei Han. 2025. Scientific paper retrieval with llm-guided semantic-based ranking. *ArXiv preprint*, abs/2505.21815.

Zhi Zheng, Kai Hui, Ben He, Xianpei Han, Le Sun, and Andrew Yates. 2020. BERT-QE: Contextualized Query Expansion for Document Re-ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4718–4728, Online. Association for Computational Linguistics.

## A Detailed illustration for hierarchical register schema

Since different types of papers have the different styles and formats, we design five kinds of hierarchical register schema, corresponding to five types of paper including algorithm innovation, benchmark construction, mechanism exploration, survey, and theory proof. We use a large language model to determine the type of input paper and then assign the corresponding hierarchical register schema. The details of each kind of schema is in Figure 6 7 8 9 10. And the instruction used to help LLM to do determine is ==*Instruction== Determine the category of the paper based on its abstract. The category options are as follows: Algorithm Innovation: Proposes new systems, new models, new training methods, new inference approaches, new data organization methods, etc. Benchmark Construction: Introduces a new benchmark. Mechanism Exploration: Investigates and analyzes the mechanisms of existing systems, algorithms, phenomena, or functionalities. Theory Proof: Proves a certain theory or formula. Survey and Review: Summarizes the research landscape of a particular field. Your final output should be only the best correct category of the paper, do not contain any other information or explanation. ==Abstract== abstract*

## B Detailed process and instructions for fine-grained content extracting

Learning from several widely recognized works, in which large language models are proven to possess reliable capabilities for content extraction (Edge et al., 2024; Li et al., 2025c, 2024b,c), PaperRegister uses a large language model as the extracting module. PaperRegister takes the node name and the text-formatted paper as input, uses instructions to guide in outputting corresponding content for the information node, and leaves it blank if the paper does not include corresponding content. In addition, PaperRegister also retrieve relevant paper text after the extracting as supplement to improve the register content. The instruction used for LLM is ==*Instruction== You are an content extraction expert, particularly skilled at extracting structured records from academic papers. Below, I need you to perform extraction based on a given schema. Please note the following: 1. Your extraction does not need to preserve the original text verbatim. You may paraphrase or summarize the content to make the extracted content more comprehensive and fluent. 2. Not all field names in the schema will have corresponding content in the paper. If you cannot find a precise match in the paper, leave that field empty. 3. Your final output must strictly adhere to the original schema in JSON format, starting with ’“‘json’ and ending with ’“‘’. ==Schema== schema ==Paper== paper*

| Dataset Name | # Corpus | # Query |
|---|---|---|
| LitSearch | 3400 | 597 |
| F.g.Search-1 | 4200 | 1,922 |
| F.g.Search-2 | 4200 | 1,922 |
| F.g.Search-3 | 4200 | 1,800 |

Table 6: Statistics of test data in experiments.

## C Detailed process and instructions for bottom-up content extracting

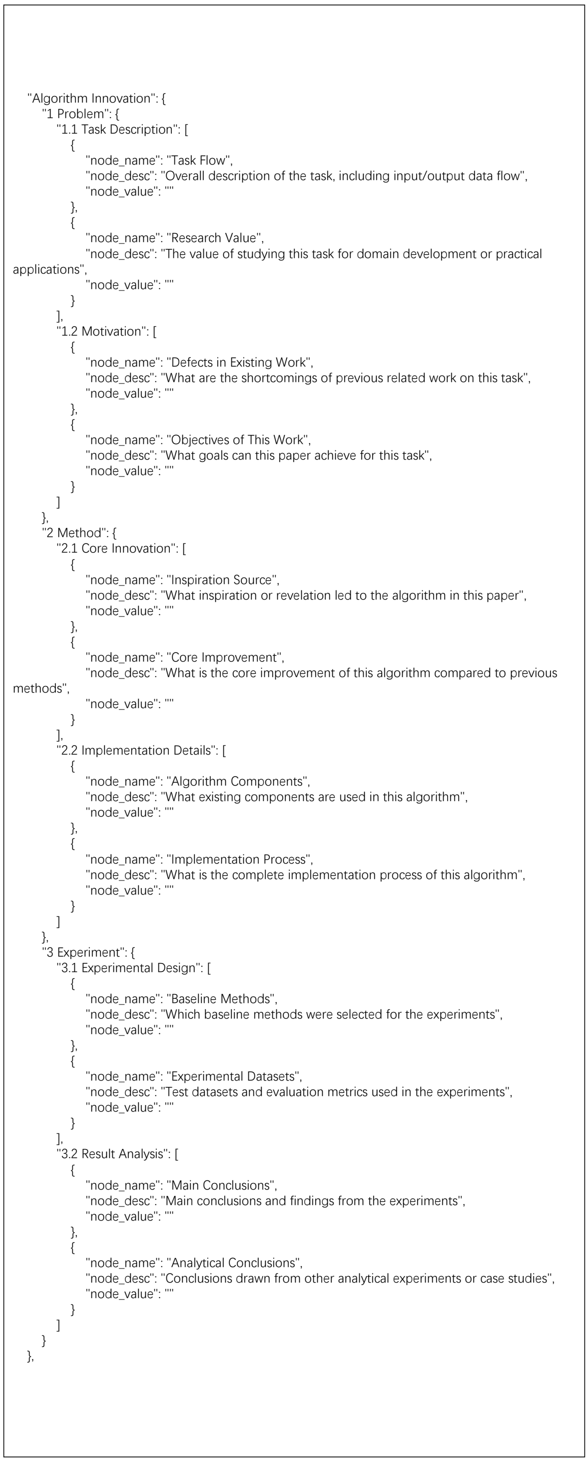
PaperRegister takes contents of sub-nodes as input and uses instructions to guide the large language model in summarizing, condensing, and removing details to turn into summary text, thus obtaining the content for upper-layer information node. In addition, PaperRegister also retrieve relevant paper text after the extracting as supplement to improve the register content. The instruction used for LLM is ==*Instruction== You are an information integration expert, and now I need your help to complete an information integration task. I will provide you with a two-level tree structure, including a root node and two child nodes. Ideally, the content of the root node should be a summary and generalization of the two child nodes. Your task is to generate the content of the root node based on the content of the two child nodes. Note the following: 1. The input I provide you is a dictionary in JSON format, including the keys ‘root_name‘ and ‘children‘, where the value of ‘children‘ is a list of child nodes. 2. Each child node contains three fields: ‘node_name‘, ‘node_desc‘, and ‘node_value‘. You should primarily use the content of ‘node_value‘ for summarization and generalization. 3. The ‘root_value‘ field should provide an abstraction and summary of the two child nodes’ contents. In other words, the root_value must not repeat keywords from the child nodes; instead, it should abstract based on those keywords. More strictly, the root_value length must not exceed that of either child node. 4. Your output should be a* *dictionary in JSON format, meaning the input dictionary will have content for the ‘root_value‘ field. The final output should start with ’“‘json’ and end with ’“‘. ==Input Tree== tree*

## D Detailed statistical information of dataset for experiments

We first use *LitSearch*, mainly containing coarse-grained queries. And then we build a new dataset, *Flexible-grained Search*, covering queries across various granularity and including data for test, development, and training. Due to limited computational resources, we are unable to conduct experiments based on an ultra-large-scale paper corpus. For LitSearch, we extract 3,400 papers as the corpus, using the original 597 paper search queries. For Flexible-grained Search, we collect 4,200 papers as the corpus. The numbers of queries included in F.g.Search-1, F.g.Search-2, and F.g.Search-3 are 1,922, 1,922, and 1,800, respectively. The statics is in Table 6.

## E Experimental results by more metrics

To provide a more comprehensive illustration of the advancements of PaperRegister over baseline methods, we present experimental results in the Table 7, 8, 9 across six metrics: Precision@5, Precision@10, MAP@5, MAP@10, NDCG@5, and NDCG@10. The data consistently support the conclusions drawn in our main experimental tables, further confirming that PaperRegister is an effective system for addressing flexible-grained paper search and the advantages of PaperRegister become more pronounced when handling finer-grained queries.



```
  "Algorithm Innovation": {
      "1 Problem": {
          "1.1 Task Description": [
              {
                  "node_name": "Task Flow",
                  "node_desc": "Overall description of the task, including input/output data flow",
                  "node_value": ""
              },
              {
                  "node_name": "Research Value",
                  "node_desc": "The value of studying this task for domain development or practical
applications",
                  "node_value": ""
              }
          ],
          "1.2 Motivation": [
              {
                  "node_name": "Defects in Existing Work",
                  "node_desc": "What are the shortcomings of previous related work on this task",
                  "node_value": ""
              },
              {
                  "node_name": "Objectives of This Work",
                  "node_desc": "What goals can this paper achieve for this task",
                  "node_value": ""
              }
          ]
      },
      "2 Method": {
          "2.1 Core Innovation": [
              {
                  "node_name": "Inspiration Source",
                  "node_desc": "What inspiration or revelation led to the algorithm in this paper",
                  "node_value": ""
              },
              {
                  "node_name": "Core Improvement",
                  "node_desc": "What is the core improvement of this algorithm compared to previous
methods",
                  "node_value": ""
              }
          ],
          "2.2 Implementation Details": [
              {
                  "node_name": "Algorithm Components",
                  "node_desc": "What existing components are used in this algorithm",
                  "node_value": ""
              },
              {
                  "node_name": "Implementation Process",
                  "node_desc": "What is the complete implementation process of this algorithm",
                  "node_value": ""
              }
          ]
      },
      "3 Experiment": {
          "3.1 Experimental Design": [
              {
                  "node_name": "Baseline Methods",
                  "node_desc": "Which baseline methods were selected for the experiments",
                  "node_value": ""
              },
              {
                  "node_name": "Experimental Datasets",
                  "node_desc": "Test datasets and evaluation metrics used in the experiments",
                  "node_value": ""
              }
          ],
          "3.2 Result Analysis": [
              {
                  "node_name": "Main Conclusions",
                  "node_desc": "Main conclusions and findings from the experiments",
                  "node_value": ""
              },
              {
                  "node_name": "Analytical Conclusions",
                  "node_desc": "Conclusions drawn from other analytical experiments or case studies",
                  "node_value": ""
              }
          ]
      }
  },
```

Figure 6: The first kind of hierarchical register schema (algorithm innovation).

| **Method** | **BM25-based Paper Search** | | | | | | | | **DPR-based Paper Search** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **LitSearch** | | **F.g.Search-1** | | **F.g.Search-2** | | **F.g.Search-3** | | **LitSearch** | | **F.g.Search-1** | | **F.g.Search-2** | | **F.g.Search-3** | |
| | P@5 | P@10 | P@5 | P@10 | P@5 | P@10 | P@5 | P@10 | P@5 | P@10 | P@5 | P@10 | P@5 | P@10 | P@5 | P@10 |
| Direct Matching | | | | | | | | | | | | | | | | |
| Title | 11.4 | 6.3 | 8.4 | 4.7 | 7.2 | 4.2 | 6.1 | 3.5 | 14.0 | 7.8 | 10.4 | 5.9 | 9.2 | 5.2 | 8.1 | 4.7 |
| Abstract | 14.1 | 7.6 | 12.7 | 6.9 | 11.5 | 6.3 | 10.8 | 6.0 | 16.4 | 8.8 | 13.9 | 7.4 | 12.4 | 6.8 | 11.6 | 6.3 |
| Total Paper | 13.5 | 7.3 | 15.9 | 8.2 | 16.3 | 8.4 | 16.8 | 8.6 | 16.0 | 8.8 | 14.6 | 7.9 | 13.8 | 7.4 | 13.2 | 7.1 |
| Query Enhancing | | | | | | | | | | | | | | | | |
| Rewriting (Ma et al., 2023) | 13.0 | 7.4 | 11.7 | 6.4 | 10.8 | 6.0 | 10.2 | 5.6 | 16.0 | 8.8 | 13.4 | 7.3 | 12.1 | 6.5 | 11.0 | 6.1 |
| HyDE (Gao et al., 2023) | **14.6** | **8.0** | 12.1 | 6.6 | 10.9 | 6.0 | 10.3 | 5.8 | 16.2 | 8.9 | 13.4 | 7.2 | 11.8 | 6.6 | 11.3 | 6.2 |
| CSQE (Lei et al., 2024) | 14.4 | 7.8 | 11.8 | 6.4 | 10.8 | 5.9 | 10.3 | 5.6 | 16.5 | 8.7 | 13.2 | 7.1 | 12.0 | 6.5 | 11.0 | 6.1 |
| Multi-field Indexing | | | | | | | | | | | | | | | | |
| Chunk$_{avg}$ (Li et al., 2025b) | 12.2 | 7.2 | 13.6 | 7.4 | 13.3 | 7.2 | 13.2 | 7.2 | 10.5 | 6.2 | 10.2 | 5.8 | 9.3 | 5.3 | 9.3 | 5.2 |
| Chunk$_{max}$ (Li et al., 2025b) | 14.3 | 7.9 | 16.0 | 8.3 | 16.3 | 8.5 | 17.1 | 8.8 | 15.2 | 8.5 | 16.0 | 8.3 | 15.3 | 8.1 | 15.8 | 8.2 |
| Paragraph$_{avg}$ (Li et al., 2025b) | 6.2 | 4.1 | 11.8 | 6.7 | 12.0 | 6.7 | 11.6 | 6.6 | 4.3 | 2.4 | 4.7 | 3.1 | 4.6 | 3.0 | 4.5 | 2.9 |
| Paragraph$_{max}$ (Li et al., 2025b) | 13.5 | 7.5 | 14.8 | 7.9 | 15.1 | 8.0 | 16.2 | 8.4 | 16.7 | 9.0 | 15.8 | 8.3 | 15.3 | 8.1 | 15.8 | 8.2 |
| Hierarchical Register Indexing | | | | | | | | | | | | | | | | |
| PaperRegister (Ours) | **14.6** | **8.0** | **17.9** | **9.1** | **17.6** | **8.9** | **17.5** | **8.9** | **17.1** | **9.3** | **16.8** | **8.7** | **16.0** | **8.3** | **16.2** | **8.3** |

Table 7: Precision@5 and Precision@10 results on paper search across various granularity.

| **Method** | **BM25-based Paper Search** | | | | | | | | **DPR-based Paper Search** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **LitSearch** | | **F.g.Search-1** | | **F.g.Search-2** | | **F.g.Search-3** | | **LitSearch** | | **F.g.Search-1** | | **F.g.Search-2** | | **F.g.Search-3** | |
| | M@5 | M@10 | M@5 | M@10 | M@5 | M@10 | M@5 | M@10 | M@5 | M@10 | M@5 | M@10 | M@5 | M@10 | M@5 | M@10 |
| Direct Matching | | | | | | | | | | | | | | | | |
| Title | 42.2 | 42.9 | 33.4 | 34.1 | 28.8 | 29.6 | 23.5 | 24.2 | 54.9 | 55.9 | 41.3 | 42.2 | 36.3 | 37.2 | 32.2 | 33.0 |
| Abstract | 52.4 | 53.0 | 54.9 | 55.6 | 49.4 | 50.1 | 45.8 | 46.5 | 65.1 | 65.9 | 59.8 | 60.5 | 51.9 | 52.7 | 47.2 | 47.8 |
| Total Paper | 51.7 | 52.4 | 74.3 | 74.6 | 76.1 | 76.4 | 79.1 | 79.3 | 60.5 | 61.6 | 63.2 | 64.0 | 59.2 | 59.9 | 55.6 | 56.3 |
| Query Enhancing | | | | | | | | | | | | | | | | |
| Rewriting (Ma et al., 2023) | 47.2 | 48.3 | 48.7 | 49.5 | 44.8 | 45.5 | 41.2 | 41.9 | 62.3 | 63.4 | 56.7 | 57.4 | 50.1 | 50.8 | 44.8 | 45.6 |
| HyDE (Gao et al., 2023) | 54.1 | 55.1 | 50.1 | 50.9 | 44.6 | 45.3 | 41.7 | 42.6 | 64.3 | 65.3 | 56.4 | 57.1 | 48.7 | 49.6 | 45.1 | 45.9 |
| CSQE (Lei et al., 2024) | 54.1 | 54.8 | 41.2 | 42.0 | 37.8 | 38.4 | 36.7 | 37.4 | 65.2 | 65.8 | 54.1 | 54.9 | 48.4 | 49.1 | 44.4 | 45.1 |
| Multi-field Indexing | | | | | | | | | | | | | | | | |
| Chunk$_{avg}$ (Li et al., 2025b) | 40.6 | 42.0 | 54.4 | 55.2 | 54.9 | 55.6 | 54.4 | 55.2 | 34.9 | 36.1 | 39.4 | 40.4 | 36.6 | 37.5 | 35.1 | 35.9 |
| Chunk$_{max}$ (Li et al., 2025b) | 56.1 | 57.1 | 73.6 | 74.0 | 75.3 | 75.8 | 79.5 | 79.8 | 62.0 | 63.0 | 71.7 | 72.2 | 68.4 | 68.9 | 69.4 | 69.9 |
| Paragraph$_{avg}$ (Li et al., 2025b) | 19.0 | 20.2 | 44.9 | 46.0 | 46.2 | 47.1 | 44.8 | 45.9 | 15.0 | 15.3 | 15.9 | 16.8 | 15.7 | 16.6 | 15.1 | 15.9 |
| Paragraph$_{max}$ (Li et al., 2025b) | 50.7 | 51.7 | 66.4 | 67.0 | 67.2 | 67.8 | 73.2 | 73.6 | 66.1 | 67.0 | 70.8 | 71.3 | 68.5 | 69.1 | 68.7 | 69.1 |
| Hierarchical Register Indexing | | | | | | | | | | | | | | | | |
| PaperRegister (Ours) | **57.4** | **58.3** | **86.2** | **86.4** | **84.9** | **85.0** | **85.3** | **85.5** | **70.0** | **70.9** | **77.4** | **77.8** | **73.7** | **74.0** | **72.9** | **73.1** |

Table 8: MAP@5 and MAP@10 results on paper search across various granularity.

| **Method** | **BM25-based Paper Search** | | | | | | | | **DPR-based Paper Search** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **LitSearch** | | **F.g.Search-1** | | **F.g.Search-2** | | **F.g.Search-3** | | **LitSearch** | | **F.g.Search-1** | | **F.g.Search-2** | | **F.g.Search-3** | |
| | N@5 | N@10 | N@5 | N@10 | N@5 | N@10 | N@5 | N@10 | N@5 | N@10 | N@5 | N@10 | N@5 | N@10 | N@5 | N@10 |
| Direct Matching | | | | | | | | | | | | | | | | |
| Title | 45.4 | 47.1 | 35.6 | 37.3 | 30.6 | 32.5 | 25.2 | 26.8 | 58.2 | 60.3 | 44.0 | 46.2 | 38.7 | 40.7 | 34.3 | 36.4 |
| Abstract | 56.3 | 57.7 | 57.1 | 58.8 | 51.4 | 53.1 | 47.9 | 49.6 | 68.5 | 70.3 | 62.2 | 63.8 | 54.5 | 56.4 | 50.0 | 51.5 |
| Total Paper | 55.0 | 56.6 | 75.6 | 76.5 | 77.5 | 78.1 | 80.3 | 81.0 | 64.4 | 66.8 | 65.7 | 67.5 | 61.6 | 63.3 | 58.2 | 59.8 |
| Query Enhancing | | | | | | | | | | | | | | | | |
| Rewriting (Ma et al., 2023) | 51.2 | 53.7 | 51.1 | 53.0 | 47.1 | 48.9 | 43.6 | 45.4 | 66.0 | 68.4 | 59.3 | 61.1 | 52.7 | 54.3 | 47.4 | 49.3 |
| HyDE (Gao et al., 2023) | 58.1 | 60.3 | 52.7 | 54.6 | 47.0 | 48.9 | 44.2 | 46.1 | 67.6 | 69.9 | 59.0 | 60.8 | 51.3 | 53.5 | 47.9 | 49.8 |
| CSQE (Lei et al., 2024) | 58.0 | 59.7 | 45.7 | 47.4 | 41.9 | 43.4 | 40.4 | 41.9 | 68.6 | 69.8 | 57.1 | 58.8 | 51.4 | 53.0 | 47.1 | 48.9 |
| Multi-field Indexing | | | | | | | | | | | | | | | | |
| Chunk$_{avg}$ (Li et al., 2025b) | 45.2 | 48.4 | 57.8 | 59.8 | 57.8 | 59.5 | 57.3 | 59.1 | 38.8 | 41.8 | 42.3 | 44.7 | 39.1 | 41.3 | 37.9 | 39.8 |
| Chunk$_{max}$ (Li et al., 2025b) | 59.2 | 61.7 | 75.2 | 76.2 | 76.8 | 77.9 | 81.0 | 81.7 | 64.9 | 67.3 | 73.8 | 74.8 | 70.5 | 71.7 | 71.8 | 72.8 |
| Paragraph$_{avg}$ (Li et al., 2025b) | 21.8 | 24.7 | 48.4 | 50.9 | 49.6 | 51.7 | 48.2 | 50.8 | 16.4 | 17.2 | 17.7 | 20.0 | 17.5 | 19.7 | 16.9 | 19.0 |
| Paragraph$_{max}$ (Li et al., 2025b) | 54.4 | 56.6 | 68.2 | 69.9 | 69.3 | 70.7 | 75.1 | 76.1 | 69.7 | 71.5 | 72.9 | 74.1 | 70.5 | 72.1 | 71.2 | 72.2 |
| Hierarchical Register Indexing | | | | | | | | | | | | | | | | |
| PaperRegister (Ours) | **60.7** | **62.9** | **87.1** | **87.5** | **85.7** | **86.0** | **85.9** | **86.3** | **73.0** | **75.0** | **79.1** | **80.1** | **75.2** | **76.1** | **74.9** | **75.5** |

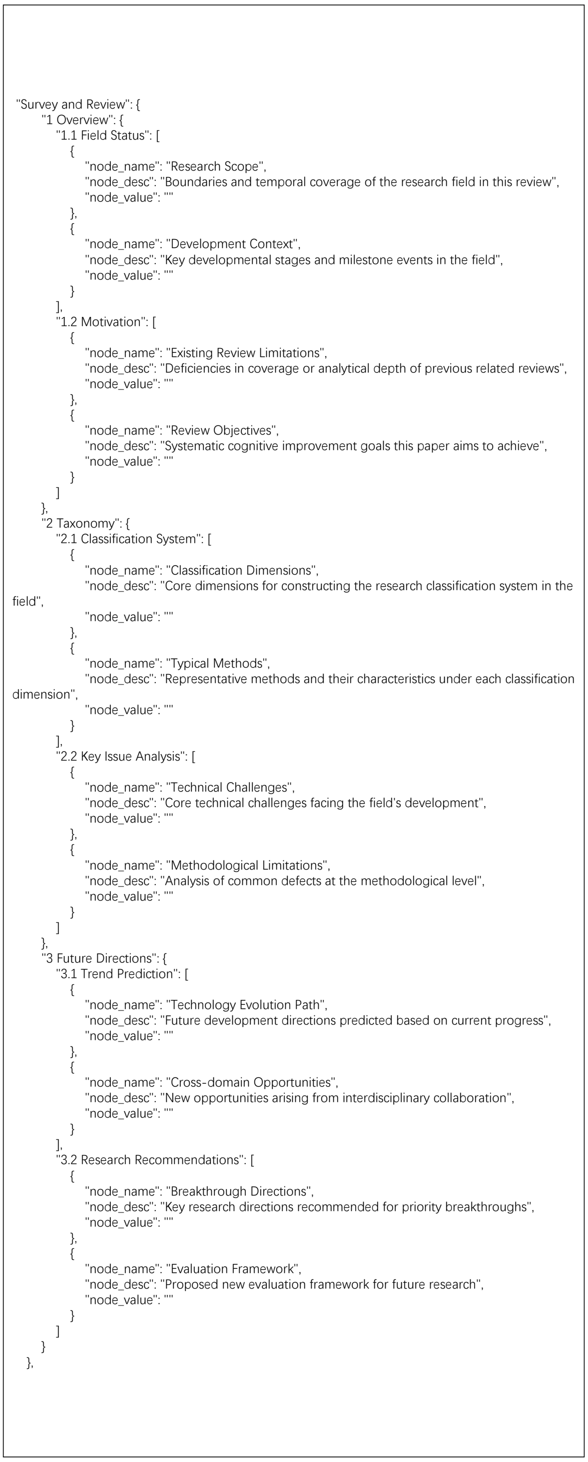
Table 9: NDCG@5 and NDCG@10 results on paper search across various granularity.

```
"Benchmark Construction": {
    "1 Problem": {
        "1.1 Task Description": [
            {
                "node_name": "Task Flow",
                "node_desc": "Input/output and data flow of the task in the benchmark",
                "node_value": ""
            },
            {
                "node_name": "Research Value",
                "node_desc": "The value of studying this task for domain development or practical
applications",
                "node_value": ""
            }
        ],
        "1.2 Motivation": [
            {
                "node_name": "Defects in Existing Benchmarks",
                "node_desc": "Various shortcomings of previous benchmarks",
                "node_value": ""
            },
            {
                "node_name": "Objectives of New Benchmark",
                "node_desc": "What goals this benchmark aims to achieve",
                "node_value": ""
            }
        ]
    },
    "2 Method": {
        "2.1 Construction Method": [
            {
                "node_name": "Data Sources",
                "node_desc": "Sources and scope of data collection for building the benchmark",
                "node_value": ""
            },
            {
                "node_name": "Annotation Scheme",
                "node_desc": "How raw data is cleaned and annotated",
                "node_value": ""
            }
        ],
        "2.2 Evaluation System": [
            {
                "node_name": "Evaluation Dimensions",
                "node_desc": "Dimensions for evaluating a system using this benchmark",
                "node_value": ""
            },
            {
                "node_name": "Evaluation Metrics",
                "node_desc": "Specific evaluation metrics and calculation methods for this benchmark",
                "node_value": ""
            }
        ]
    },
    "3 Experiment": {
        "3.1 Experimental Design": [
            {
                "node_name": "Selected Methods for Testing",
                "node_desc": "Which existing methods were tested using this benchmark",
                "node_value": ""
            },
            {
                "node_name": "Experimental Settings",
                "node_desc": "Specific settings for the testing",
                "node_value": ""
            }
        ],
        "3.2 Result Analysis": [
            {
                "node_name": "Main Conclusions",
                "node_desc": "Main conclusions drawn from testing with this benchmark",
                "node_value": ""
            },
            {
                "node_name": "Analytical Conclusions",
                "node_desc": "Conclusions drawn from other analytical experiments or case studies",
                "node_value": ""
            }
        ]
    }
  },
```

Figure 7: The second kind of hierarchical register schema (benchmark construction).

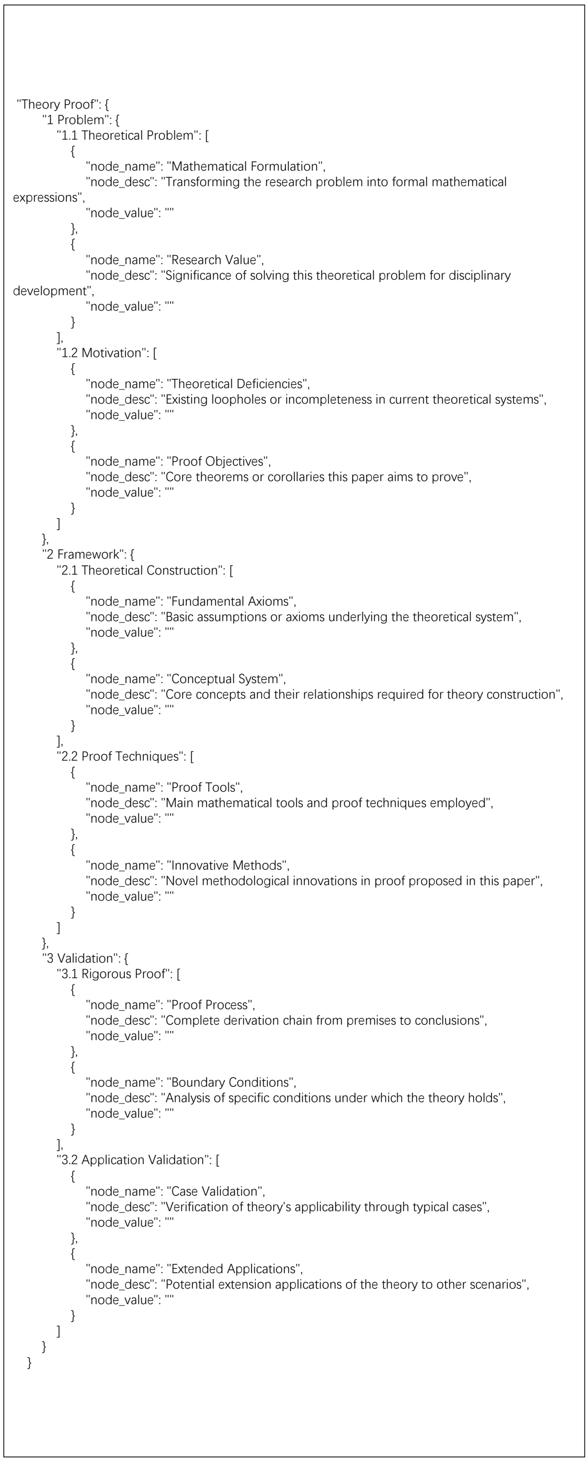
```
"Mechanism Exploration": {
    "1 Problem": {
        "1.1 Research Subject": [
            {
                "node_name": "Scientific Problem",
                "node_desc": "What is the problem this work aims to explore",
                "node_value": ""
            },
            {
                "node_name": "Research Value",
                "node_desc": "What is the importance of exploring this problem",
                "node_value": ""
            }
        ],
        "1.2 Motivation": [
            {
                "node_name": "Defects in Existing Work",
                "node_desc": "What limitations exist in previous exploration work",
                "node_value": ""
            },
            {
                "node_name": "Objectives of This Work",
                "node_desc": "What conclusions or goals this paper expects to achieve",
                "node_value": ""
            }
        ]
    },
    "2 Method": {
        "2.1 Practical Exploration": [
            {
                "node_name": "Components",
                "node_desc": "What existing methods, data, or models are used in the exploration
process",
                "node_value": ""
            },
            {
                "node_name": "Implementation Process",
                "node_desc": "What process is used to achieve the exploration objectives of this work",
                "node_value": ""
            }
        ],
        "2.2 Theoretical Derivation": [
            {
                "node_name": "Fundamental Theories",
                "node_desc": "What fundamental theories underlie this derivation process",
                "node_value": ""
            },
            {
                "node_name": "Derivation Process",
                "node_desc": "Specific process of theoretical derivation",
                "node_value": ""
            }
        ]
    },
    "3 Experiment": {
        "3.1 Conclusion Presentation": [
            {
                "node_name": "Exploration Conclusions",
                "node_desc": "Main conclusions drawn about the explored problem in the paper",
                "node_value": ""
            },
            {
                "node_name": "Supporting Evidence",
                "node_desc": "Specific evidence supporting these conclusions",
                "node_value": ""
            }
        ],
        "3.2 Guidance": [
            {
                "node_name": "Technical Improvement Directions",
                "node_desc": "Recommendations for technical development based on the paper's
conclusions",
                "node_value": ""
            },
            {
                "node_name": "Other Discussions",
                "node_desc": "Other discussions and suggestions for research in this field",
                "node_value": ""
            }
        ]
    }
  },
```

Figure 8: The third kind of hierarchical register schema (mechanism exploration).



```
"Survey and Review": {
    "1 Overview": {
        "1.1 Field Status": [
            {
                "node_name": "Research Scope",
                "node_desc": "Boundaries and temporal coverage of the research field in this review",
                "node_value": ""
            },
            {
                "node_name": "Development Context",
                "node_desc": "Key developmental stages and milestone events in the field",
                "node_value": ""
            }
        ],
        "1.2 Motivation": [
            {
                "node_name": "Existing Review Limitations",
                "node_desc": "Deficiencies in coverage or analytical depth of previous related reviews",
                "node_value": ""
            },
            {
                "node_name": "Review Objectives",
                "node_desc": "Systematic cognitive improvement goals this paper aims to achieve",
                "node_value": ""
            }
        ]
    },
    "2 Taxonomy": {
        "2.1 Classification System": [
            {
                "node_name": "Classification Dimensions",
                "node_desc": "Core dimensions for constructing the research classification system in the
field",
                "node_value": ""
            },
            {
                "node_name": "Typical Methods",
                "node_desc": "Representative methods and their characteristics under each classification
dimension",
                "node_value": ""
            }
        ],
        "2.2 Key Issue Analysis": [
            {
                "node_name": "Technical Challenges",
                "node_desc": "Core technical challenges facing the field's development",
                "node_value": ""
            },
            {
                "node_name": "Methodological Limitations",
                "node_desc": "Analysis of common defects at the methodological level",
                "node_value": ""
            }
        ]
    },
    "3 Future Directions": {
        "3.1 Trend Prediction": [
            {
                "node_name": "Technology Evolution Path",
                "node_desc": "Future development directions predicted based on current progress",
                "node_value": ""
            },
            {
                "node_name": "Cross-domain Opportunities",
                "node_desc": "New opportunities arising from interdisciplinary collaboration",
                "node_value": ""
            }
        ],
        "3.2 Research Recommendations": [
            {
                "node_name": "Breakthrough Directions",
                "node_desc": "Key research directions recommended for priority breakthroughs",
                "node_value": ""
            },
            {
                "node_name": "Evaluation Framework",
                "node_desc": "Proposed new evaluation framework for future research",
                "node_value": ""
            }
        ]
    }
  },
```

Figure 9: The fourth kind of hierarchical register schema (survey and review).



```
"Theory Proof": {
    "1 Problem": {
        "1.1 Theoretical Problem": [
            {
                "node_name": "Mathematical Formulation",
                "node_desc": "Transforming the research problem into formal mathematical
expressions",
                "node_value": ""
            },
            {
                "node_name": "Research Value",
                "node_desc": "Significance of solving this theoretical problem for disciplinary
development",
                "node_value": ""
            }
        ],
        "1.2 Motivation": [
            {
                "node_name": "Theoretical Deficiencies",
                "node_desc": "Existing loopholes or incompleteness in current theoretical systems",
                "node_value": ""
            },
            {
                "node_name": "Proof Objectives",
                "node_desc": "Core theorems or corollaries this paper aims to prove",
                "node_value": ""
            }
        ]
    },
    "2 Framework": {
        "2.1 Theoretical Construction": [
            {
                "node_name": "Fundamental Axioms",
                "node_desc": "Basic assumptions or axioms underlying the theoretical system",
                "node_value": ""
            },
            {
                "node_name": "Conceptual System",
                "node_desc": "Core concepts and their relationships required for theory construction",
                "node_value": ""
            }
        ],
        "2.2 Proof Techniques": [
            {
                "node_name": "Proof Tools",
                "node_desc": "Main mathematical tools and proof techniques employed",
                "node_value": ""
            },
            {
                "node_name": "Innovative Methods",
                "node_desc": "Novel methodological innovations in proof proposed in this paper",
                "node_value": ""
            }
        ]
    },
    "3 Validation": {
        "3.1 Rigorous Proof": [
            {
                "node_name": "Proof Process",
                "node_desc": "Complete derivation chain from premises to conclusions",
                "node_value": ""
            },
            {
                "node_name": "Boundary Conditions",
                "node_desc": "Analysis of specific conditions under which the theory holds",
                "node_value": ""
            }
        ],
        "3.2 Application Validation": [
            {
                "node_name": "Case Validation",
                "node_desc": "Verification of theory's applicability through typical cases",
                "node_value": ""
            },
            {
                "node_name": "Extended Applications",
                "node_desc": "Potential extension applications of the theory to other scenarios",
                "node_value": ""
            }
        ]
    }
  }
```

Figure 10: The fifth kind of hierarchical register schema (theory proof).