FlowSwitch: A State-Aware Framework for Workflow Transitions in Adaptive Dialogue Agents

Wen-Yu Chang[1] **Luning Qiu**[2] **Yi-Hung Liu**[1] **Yun-Nung Chen**[1]
[1]National Taiwan University, Taipei, Taiwan
[2]University of Science and Technology Beijing, Beijing, China
f10946031@csie.ntu.edu.tw y.v.chen@ieee.org

Abstract

Integrating workflow knowledge into large language models (LLMs) is essential for enabling real-world task-solving capabilities. However, real-world conversations are inherently dynamic; users frequently shift intents or request actions beyond the scope of the current workflow. Existing systems often fail to robustly detect such transitions or determine when to retrieve a new workflow. This paper presents FlowSwitch, a state-aware framework designed to dynamically manage workflow transitions during multi-turn dialogues. FlowSwitch functions as a decision-making agent that autonomously determines whether to continue the current path or query for new workflow knowledge based on contextual representations. When a search is triggered, a dedicated retriever identifies the most relevant workflow knowledge. Comprehensive experiments on workflow representation and retrieval strategies demonstrate that FlowSwitch achieves superior retrieval performance, particularly when leveraging agent-generated search queries. Furthermore, our framework reduces search operations by nearly 50%, significantly lowering computational costs and response latency.[1]

1 Introduction

Task-oriented dialogue (ToD) systems have become ubiquitous across diverse domains, ranging from customer service and personal assistance to e-commerce and logistics (Budzianowski et al., 2018; Rastogi et al., 2019). However, real-world applications increasingly demand capabilities that extend beyond single-task execution (Sun et al., 2016a,b; Kuo and Chen, 2023). For instance, a restaurant service bot must seamlessly handle a user who first explicitly inquires about table availability, then modifies an existing reservation, and subsequently asks about menu allergens. Each of these requests corresponds to a distinct workflow with specific procedural steps and data requirements. Consequently, the ability to detect when a user's intent drifts beyond the scope of the current workflow and to transition to the appropriate alternative is essential for maintaining conversational coherence.

Large language models (LLMs) have driven a shift toward adaptive agent frameworks capable of multi-step reasoning and planning (Yao et al., 2023; Qin et al., 2024). Yet, the majority of existing research focuses on optimizing workflow *execution* after a path has been selected (Zhang et al., 2024; Xiao et al., 2024). The critical question of how agents should learn to autonomously detect and manage workflow transitions during dynamic conversations remains unexplored. Traditional intent classification, which relies on predefined intent sets, is ill-suited for scenarios where workflows share semantic similarities or where user needs evolve mid-conversation (Liu and Lane, 2016; Chen et al., 2016). While recent studies address workflow planning (Xiao et al., 2024; Tan et al., 2025) and procedural compliance (Shi et al., 2025), they do not directly tackle the mechanisms of switching. This is a significant gap, as prior work demonstrates that task-switching substantially degrades LLM performance absent specific mitigation strategies (Gupta et al., 2024).

In this work, we identify two core challenges in developing robust workflow-guided dialogue systems. First, agents require **state-aware transition detection** to recognize when the dialogue state has drifted beyond the active workflow's boundaries. This requires going beyond static intent classification to leverage dialogue representations that capture both conversation history and the status of partially completed procedural steps. Second, agents need **adaptive workflow retrieval** to efficiently select the correct target workflow from large-scale libraries. Given that workflows are typically organized hierarchically by domain, role, and scenario,

[1] Code: https://github.com/MiuLab/FlowSwitch.

Proceedings of the 16th International Workshop on Spoken Dialogue Systems Technology, pages 18–33
February 26–March 1, 2026. ©2026 Association for Computational Linguistics

this structure naturally motivates the use of hierarchical retrieval strategies to enhance both efficiency and precision compared to flat retrieval approaches.

To address these challenges, we propose **FlowSwitch**, an agentic framework for state-aware workflow transitions. FlowSwitch integrates dynamic transition detection with hierarchical workflow retrieval, leveraging the dialogue context to identify transition points rather than relying on heuristic rules. We systematically evaluate retrieval strategies across workflow representations, contextual inputs, and retrieval architectures. Specifically, we compare: (1) diverse workflow representations, including *text*, *summary*, *flowchart*, and *code*; (2) contextual inputs ranging from the full dialogue history to recent turns; and (3) retrieval architectures spanning flat, two-layer, and three-layer hierarchical designs. Comprehensive experiments are conducted on 51 workflows across five real-world domains, utilizing both sparse (BM25) and dense (e5) retrievers.

Our contributions are 3-fold:

- We propose FlowSwitch, a framework that unifies workflow transition detection and hierarchical retrieval, achieving over 90% accuracy in maintaining correct workflows and a 56% reduction in search costs.
- We systematically compare retrieval strategies across various workflow formats, contextual inputs, and retriever types, demonstrating that hierarchical retrieval significantly outperforms flat approaches in complex environments.
- We provide an in-depth analysis of how structured (e.g., *code*, *flowchart*) versus semantic (e.g., *text*, *summary*) representations influence retrieval effectiveness, offering actionable guidance for building scalable, multi-workflow dialogue agents.

2 Related Work

2.1 Task-Oriented Dialogue (ToD) Systems

ToD systems have evolved substantially with the introduction of large-scale, multi-domain datasets. MultiWOZ (Budzianowski et al., 2018) contains 10,000 dialogues across seven domains, demonstrating that users naturally switch between tasks, such as booking a hotel followed by a restaurant reservation. The schema-guided dialogue (SGD) dataset (Rastogi et al., 2019) extends this scale to over 16,000 dialogues spanning 16 domains, promoting a schema-based paradigm that facilitates zero-shot transfer. While these benchmarks highlight the ubiquity of multi-domain interactions, standard baselines primarily focus on dialogue state tracking (DST) within a fixed schema rather than explicit workflow switching.

Traditional DST methods, such as TRADE (Wu et al., 2019) and SUMBT (Lee et al., 2019), focus on tracking user goals across turns by generating slot values or matching slot-utterance pairs. Although these approaches handle state updates effectively, they typically assume a fixed set of domains and lack mechanisms to determine when to switch between fundamentally different workflows. Even recent approaches applying chain-of-thought (CoT) reasoning to DST (Xu et al., 2024) focus on state accuracy rather than detecting workflow boundaries in dynamic environments.

2.2 Workflow-Guided Planning and Execution

With the rise of LLM-based agents, research has pivoted toward workflow execution and planning. ProAgent (Zhang et al., 2024) introduces agentic process automation, utilizing specialized agents for control flow and data handling. Meta-agent-workflow (Tan et al., 2025) focuses on constructing reusable workflows from LLM execution traces. Similarly, FlowBench (Xiao et al., 2024) benchmarks agents on their ability to adhere to predefined procedures across 51 scenarios. While these systems demonstrate strong capabilities in *executing* a selected workflow, they offer limited insight into detecting when to *abandon* or *switch* workflows in response to shifting user intent.

FlowAgent (Shi et al., 2025) attempts to address out-of-workflow queries by distinguishing between compliant and flexible handling modes. However, their selection mechanism relies on predefined mappings rather than dynamic retrieval, limiting scalability. Our work complements these execution-focused systems by addressing the critical upstream challenges: *detecting transition points* and *retrieving the correct workflow* from a large, unmapped workflow library.

2.3 Dialogure Retrieval Methods

Dense retrieval has become a cornerstone of information-seeking tasks. Dense passage retrieval (DPR) (Karpukhin et al., 2020) leverages dual encoders for semantic matching, while ColBERT (Khattab and Zaharia, 2020) optimizes efficiency via late interaction. In dia-

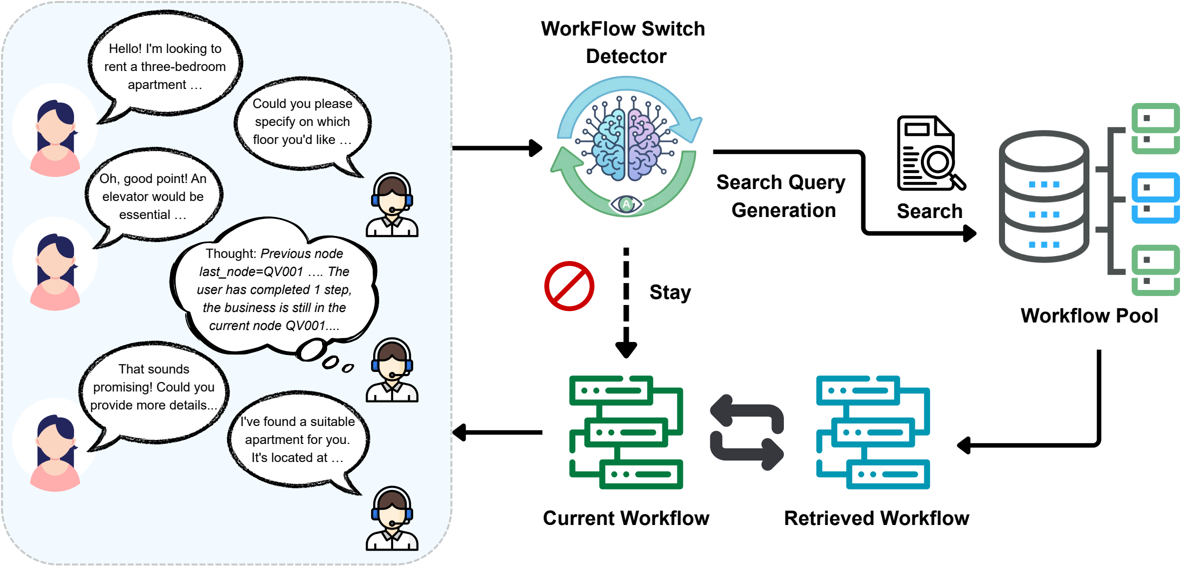
WorkFlow Switch Detector
Hello! I'm looking to rent a three-bedroom apartment ...
Could you please specify on which floor you'd like ...
Oh, good point! An elevator would be essential ...
Thought: Previous node last_node=QV001 .... The user has completed 1 step, the business is still in the current node QV001....
That sounds promising! Could you provide more details...
I've found a suitable apartment for you. It's located at ...
Search Query Generation
Search
Workflow Pool
Stay
Current Workflow
Retrieved Workflow

Figure 1: Overview of the proposed FLOWSWITCH framework.

logue contexts, retrieval-augmented generation (RAG) (Lewis et al., 2020) grounds responses in external knowledge, with recent variations like UniMS-RAG (Wang et al., 2024) dynamically selecting between multiple knowledge sources based on query characteristics.

Given the structured nature of workflows, which are often organized by domain, role, and scenario, hierarchical retrieval strategies offer significant promise. Approaches like dense hierarchical retrieval (Liu et al., 2021) and hybrid hierarchical retrieval (Arivazhagan et al., 2023) have shown that multi-stage search (e.g., document-level followed by passage-level) improves recall and zero-shot performance. We investigate whether this hierarchical intuition holds for workflow retrieval, systematically analyzing under what conditions hierarchical methods outperform flat retrieval architectures.

2.4 Transition Detection for Task Switching

The specific problem of detecting task transitions in dialogue remains under-explored. Gupta et al. (2024) provided the first systematic analysis of task-switching effects, demonstrating that LLM performance degrades significantly when tasks change mid-conversation. While their work establishes the severity of the problem, it does not propose specific detection mechanisms. Similarly, out-of-scope (OOS) detection methods (Zhan et al., 2021; Zaera et al., 2025) identify when requests fall outside system capabilities, but they do not resolve *which* alternative workflow should be engaged.

Conventional intent classification approaches track intent changes but typically assume finite, predefined intent sets mapped to static handlers (Liu and Lane, 2016; Goo et al., 2018; Liu et al., 2024). These methods struggle to scale to scenarios with numerous, semantically overlapping workflows. We depart from this classification-based paradigm by framing workflow selection as a retrieval problem, enabling systems to scale to large libraries and leverage the natural hierarchical organization of tasks.

3 Methodology

3.1 Problem Formalization

We consider a task-oriented dialogue agent equipped with a library of workflows $\mathcal{W} = \{w_1, w_2, \ldots, w_N\}$, where each workflow w_i encapsulates the procedural logic for a specific domain or task. At any turn t, the system's state is defined by the cumulative dialogue history H_t and the currently active workflow $w_a \in \mathcal{W}$.

The core challenge is to dynamically detect when the user's intent drifts beyond the scope of w_a. We formalize this as a binary decision process governed by a transition policy π_{switch}:

$$\pi_{\text{switch}}(H_t, w_a) \in \{\textsc{Stay}, \textsc{Search}\}.$$

This policy evaluates whether the current workflow w_a remains valid given the context H_t.

- **Search Mode:** If $\pi_{\text{switch}}(H_t, w_a) = \textsc{Search}$, the system invokes a retriever R_ϕ parameterized by ϕ (e.g., a sparse or dense encoder). The retriever queries the library $\mathcal{W}$ to identify the workflow w^* that best aligns with the updated state:

$$w^* = R_\phi(H_t, \mathcal{W}).$$

The active workflow is then updated to w^*, and the agent proceeds with the new procedure.

- **Stay Mode:** If $\pi_{\text{switch}}(H_t, w_a) = \textsc{Stay}$, the agent retains w_a as the active workflow and executes the next procedural step.

The overall objective is to maximize conversational coherence by ensuring the active workflow consistently matches the evolving user intent.

3.2 FLOWSWITCH Framework

Our framework empowers dialogue agents to dynamically manage workflow adherence and transitions during multi-turn interactions. As illustrated in Figure 1, FLOWSWITCH comprises two primary components: (1) a **Workflow Switch Detector**, which continuously evaluates whether the current workflow remains valid given the evolving context, and (2) an **Adaptive Retriever**, which identifies and activates the optimal target workflow when a transition is deemed necessary.

3.3 Retrieval Pool Construction

We construct our library using FlowBench (Xiao et al., 2024), a dataset of diverse, workflow-guided conversations. The dataset spans 51 workflows organized hierarchically into 22 roles across 6 domains. To leverage this inherent structure, we construct three distinct levels of retrieval pools:

- **Domain Pool:** We aggregate all roles within a specific domain and prompt an LLM to synthesize a high-level description summarizing the domain's collective functionality.
- **Role Pool:** Similarly, for each role, we generate a concise description based on the specific set of workflows associated with that role.
- **Workflow Pool:** At the granular workflow level, we maintain four distinct representations for each scenario: *text*, *code*, *flowchart*, and *summary*. The first three are extracted directly from the dataset, while the *summary* is generated via an LLM to provide a compact semantic abstraction.

For instance, in the *Customer Service* domain, roles such as *restaurant_waiter* and *apartment_manager* include workflows like *[Restaurant Search, Restaurant Booking]* and *[Apartment Search, Schedule a Viewing]*.

3.4 Workflow Switch Detector

The workflow switch detector serves as the framework's decision-making core. At every turn, it monitors the dialogue for intent shifts relative to the active workflow w_a. We implement this module using an LLM that processes the current dialogue history H_t and the active workflow content.

The detector operates as a dual-function module:

1. **Decision:** It outputs a binary decision, π_{switch}. If the user's intent remains within scope, it outputs $\textsc{Stay}$, and the agent continues executing w_a without initiating retrieval.
2. **Query Generation:** If the detector predicts a shift (outputting $\textsc{Search}$), it simultaneously generates a structured search query. This query contains: (1) potential workflow names, (2) a target task description, and (3) the expected next action.

We hypothesize that this *self-generated query* is rich in semantic context and tailored to the agent's immediate needs, providing significantly stronger retrieval signals than the raw dialogue history.

3.5 Retrieval Strategies

We investigate two categories of retrieval strategies: (1) embedding-based retrieval (without LLM inference), and (2) LLM-guided hierarchical retrieval. Both strategies exploit the multi-level structure of our pools to optimize search efficiency and relevance.

(1) Embedding-Based Retrieval In this setting, retrieval relies solely on vector similarity without intermediate LLM reasoning.

- **Flat Retrieval:** The retriever searches the entire workflow pool directly. Given an input query, it returns the top-k workflows with the highest similarity scores based on the chosen method R_ϕ.
- **Hierarchical Retrieval:** To constrain the search space, the retriever first identifies the most relevant *domain* or *role* from the higher-level pools. It then restricts the subsequent workflow search to the subset of scenarios associated with that selected domain or role, improving precision by filtering out irrelevant categories early.

(2) LLM-Guided Hierarchical Retrieval Here, an LLM acts as a semantic router, selecting candidate domains or roles before the embedding-based

Retrieval Strategy	**LLM-Guided**	**Top-1**	**Top-2**	**Top-3**	**Top-4**	**Top-5**	**MAP**
Without Workflow Switch Detector							
Flat Retrieval	✗	37.3	55.3	62.8	66.1	68.7	52.2
Hierarchical (**D**→ **W**)	✗	32.9	52.0	58.7	62.0	64.1	47.8
	✓	39.5	54.1	60.4	63.8	66.4	51.3
Hierarchical (**D**→ **R**→ **W**)	✗	33.1	51.7	58.5	61.9	64.1	47.8
	✓	41.7	60.5	67.6	71.5	73.1	56.3
Hierarchical (**R**→ **W**)	✗	37.6	57.1	65.8	70.2	72.1	53.7
	✓	41.8	61.3	69.2	73.2	74.4	57.1
With Workflow Switch Detector							
Flat Retrieval	✗	63.8	71.3	74.8	76.6	78.0	78.5
Hierarchical (**D**→ **W**)	✗	62.6	70.6	73.8	75.7	76.7	76.8
	✓	64.8	71.6	74.7	76.4	77.7	78.7
Hierarchical (**D**→ **R** → **W**)	✗	62.7	70.5	73.8	75.8	76.9	76.8
	✓	65.5	74.6	78.1	80.1	81.2	80.9
Hierarchical (**R**→ **W**)	✗	64.2	72.1	76.4	78.9	80.3	79.1
	✓	**65.6**	**74.8**	**78.9**	**81.1**	**82.0**	**81.4**

Table 1: Workflow retrieval results w/o & w/ workflow switch detector usingE5 retriever in different settings (%).

search occurs. We explore three hierarchical configurations:

- **Two-Layer (Domain → Scenario):** The LLM analyzes the dialogue context to select the most relevant domains. The retriever then ranks workflows solely within those domains to produce the top-k candidates.
- **Two-Layer (Role → Scenario):** The LLM selects the most relevant roles directly. The retriever then searches the workflows associated with these roles.
- **Three-Layer (Domain → Role → Scenario):** This approach fully mirrors the dataset's hierarchy. The LLM first selects top domains, then identifies specific roles within those domains, and finally, the retriever ranks the workflows under those roles. This multi-step reasoning allows for finer-grained filtering compared to broader domain-level selection.

4 Experiments

4.1 Implementation Details

In the experiments, We employ `Qwen3-14B` (Yang et al., 2025) as the backbone for the workflow switch detector and `Qwen3-8B` as the hierarchical router to select *domains* and *roles* in the LLM-guided settings. For the final workflow retrieval, we adopt `E5-base-v2` (Wang et al., 2022) as the dense embedding model.

Regarding retrieval hyperparameters, we select the top-$k = 3$ candidates for the domain and role levels, and the top-$k = 5$ candidates for the final workflow level. For a complete list of hyperparameters, please refer to Appendix A.

4.2 Datasets and Evaluation

Our experiments utilize FlowBench (Xiao et al., 2024), which contains 2,219 turn-level samples across 51 real-world task scenarios. Each workflow in the dataset includes three native representations: *text*, *code*, and *flowchart*. To augment this, we employ `GPT-4.1` to generate a fourth representation, *summary*, providing a concise semantic abstraction. The same model is used to generate the high-level descriptions for the *domain* and *role* pools.

We evaluate retrieval performance using Top-k accuracy and mean average precision (MAP). To isolate retrieval effectiveness, we first benchmark performance *without* the switch detector (i.e., forcing retrieval at every turn). We compare four primary strategies: (1) *Naive Flat Retrieval*, (2) *Domain → Workflow*, (3) *Role → Workflow*, and (4) *Domain → Role → Workflow*. Each hierarchical strategy is evaluated under two conditions: purely embedding-based versus LLM-guided at the upper layers.[2]

4.3 Results

We evaluate FLOWSWITCH on the FlowBench turn-level benchmark, reporting Top-k accuracy and

[2] Results are averaged across all workflow formats and query types unless noted otherwise. Comprehensive breakdowns for each configuration can be found in Appendix C.

Retriever	Top-1	MAP
Without Workflow Switch Detector		
E5	40.0	54.0
BM25	42.0	52.0
With Workflow Switch Detector		
-	**90.5**	**88.9**

Table 2: Workflow decision performance when detector decides to STAY (%).

Query	Top-1	Top-2	Top-3	Top-4	Top-5	MAP
Full	61.8	70.7	74.4	**76.7**	**78.0**	76.6
Last3	63.2	70.6	74.2	76.4	77.6	77.4
Last2	63.8	**71.0**	**74.5**	76.4	77.5	77.9
Last1	64.2	70.8	73.9	75.8	77.0	77.8
Self-Gen	**65.4**	70.8	73.4	75.1	76.3	**78.7**

Table 3: Retrieval performance w/ the workflow switch detector averaged over all retrieval strategies (%).

mean average precision (MAP) across various retrieval strategies. Table 1 summarizes the performance averaged across all formats and query formulations. We focus on results using the denseE5 retriever; while trends with BM25 are consistent, absolute scores are uniformly lower (Appendix C). The results indicate that hierarchical retrieval utilizing intermediate role information yields superior performance. Specifically, the LLM-guided *Role→Workflow* configuration achieves the highest metrics, reaching Top-1 of 65.6% and MAP of 81.4%. Crucially, integrating the workflow switch detector significantly outperforms the baseline strategy of performing retrieval at every turn across all configurations.

Beyond accuracy, FLOWSWITCH delivers substantial efficiency gains. By effectively filtering out redundant searches, the switch detector not only lowers computational costs but also drives massive improvements over continuous-retrieval baselines (e.g., boosting flat retrieval Top-1 from ≈37% to 64%). These findings confirm that coupling a robust, state-aware switch policy with hierarchical retrieval effectively balances high precision with low computational overhead.

5 Discussion

Efficiency and Effectiveness of the Workflow Switch Detector. To further evaluate the workflow switch detector, we analyze its accuracy specifically when the correct decision is to STAY (i.e., refrain from searching). In these scenarios, the ground truth corresponds to either the currently active workflow or an empty assignment (indicating no transition is required). As detailed in Table 2, disabling the detector forces the system to perform retrieval at every turn. This baseline approach yields poor results: the dense (E5) and lexical (BM25) retrievers achieve only 40% and 42% Top-1 accuracy, respectively, with MAP scores hovering around 52–54%. In contrast, enabling the Workflow Switch Detector dramatically boosts performance, achieving a Top-1 accuracy of **90.5%** and a MAP of **88.9%**. This confirms the module's robust ability to maintain workflow continuity. On average, the detector correctly suppresses retrieval in **1,244.5** turns[3], maintaining a **90%** decision accuracy and delivering a **56% reduction** in total search operations. These results demonstrate that the detector acts as a critical filter, enhancing both retrieval precision and computational efficiency by eliminating redundant searches during multi-turn dialogues.

Impact of Query Formulation. As detailed in Table 3, the self-generated *search query* achieves the highest performance in terms of Top-1 accuracy and MAP. However, for broader metrics (Top-2 to Top-5), using the full dialogue context remains competitive. We hypothesize this is because the full history contains the agent's prior actions, which naturally overlap with the procedural details of valid workflows. This lexical overlap can inflate similarity scores, helping the correct workflow appear in the Top-k candidates even if it is not ranked first.

To isolate the true discriminative power of the inputs, we analyze the challenging subset where the ground-truth workflow explicitly differs from the current active workflow, a scenario requiring the system to break context inertia. As illustrated in Figures 2, the *search query* input consistently outperforms raw context inputs across both retriever types in this regime, exhibiting significantly lower variance. These results highlight that a targeted, semantically grounded query is essential for accurately navigating workflow transitions.

Impact of Workflow Representation. To account for real-world data heterogeneity, our main results average performance across four workflow formats: *code*, *flowchart*, *text*, and *summary*. A granular analysis (Table 4) reveals that structured representations, specifically *code* and *flowchart*, yield supe-

[3] Averaged across the four workflow formats: *text*, *code*, *flowchart*, and *summary*.

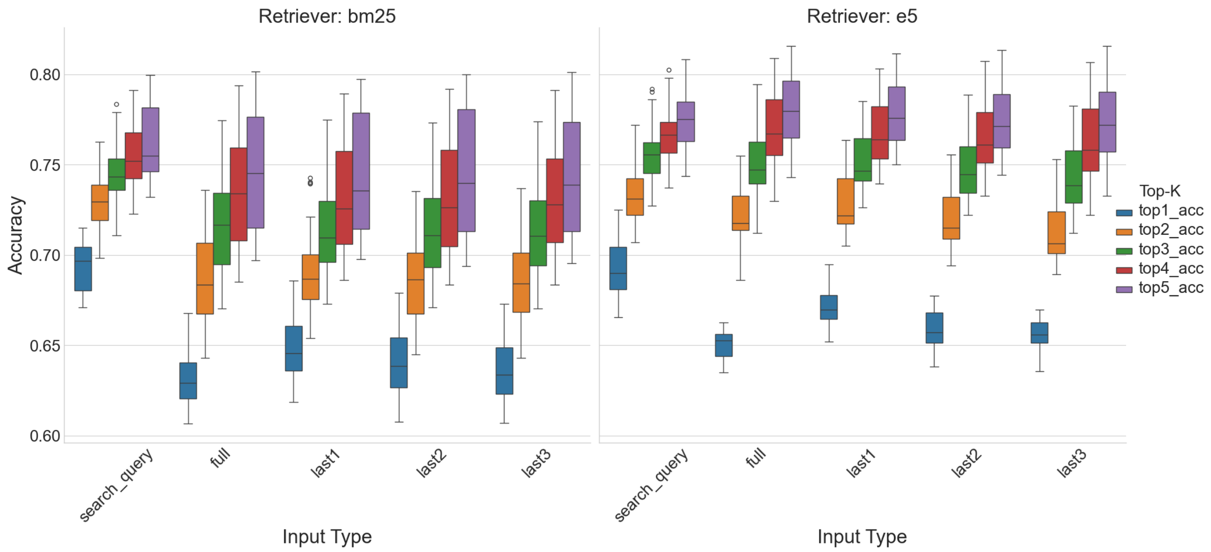

Figure 2: Top-k accuracy across different search query types.

Input Type	LLM-Guided	Retriever	Top-1	Top-2	Top-3	Top-4	Top-5	MAP
Text	✗	BM25	60.8	67.2	70.2	71.9	73.2	74.1
	✗	E5	62.1	70.3	73.9	76.0	77.3	77.1
	✓	BM25	64.3	71.9	75.5	77.7	79.0	79.6
	✓	E5	63.0	71.6	76.0	78.1	79.3	78.5
Code	✗	BM25	61.6	66.8	69.5	71.2	72.4	73.6
	✗	E5	63.2	71.0	74.7	76.9	78.3	77.6
	✓	BM25	64.0	70.6	74.1	76.4	77.8	78.0
	✓	E5	67.1	75.3	**78.9**	**80.7**	**81.8**	**82.6**
Flowchart	✗	BM25	64.4	70.6	72.8	73.9	747	77.3
	✗	E5	64.7	72.0	75.4	77.2	784	78.8
	✓	BM25	66.7	73.6	77.0	78.7	79.7	81.3
	✓	E5	**67.2**	**75.8**	78.7	80.3	81.0	82.4
Summary	✗	BM25	62.0	67.9	70.6	72.7	73.8	74.4
	✗	E5	63.4	71.2	74.8	76.8	78.0	77.6
	✓	BM25	62.6	68.0	71.6	74.6	76.7	75.7
	✓	E5	64.0	71.9	75.4	77.7	79.2	77.8

Table 4: Retrieval performance of different workflow formats in all settings with the workflow switch detector (%).

Workflow Format	# of Stay	# of Search
Text	1,209	1,010
Code	1,231	988
Flowchart	1,276	943
Summary	1,262	957

Table 5: SEARCH and STAY decision counts of the workflow switch detector across workflow formats.

rior performance, achieving MAP scores exceeding 82% with dense retrievers. We attribute this to the explicit procedural cues inherent in these formats; they clearly delineate logic steps and transitions, allowing the retriever to anchor queries more effectively. In contrast, *text* and *summary* formats, while semantically rich, suffer from higher ambiguity and overlap between similar workflows, resulting in slightly lower accuracy. Consistent with prior trends, dense retrievers (E5) outperform lexical baselines (BM25) across all formats, with LLM assistance further boosting recall. These findings suggest that for complex task-oriented systems, retaining the native structural properties of workflows provides a stronger retrieval signal than flattening them into natural language descriptions.

Finally, Table 5 compares the distribution of SEARCH vs. STAY decisions across these formats. The decision counts remain highly consistent regardless of the underlying workflow representation. This implies that while the format significantly influences the retriever's ability to find the correct target, it has negligible impact on the detector's ability to recognize when a transition is needed.

Impact of Hierarchical Categorization. Leveraging hierarchical structure, particularly at the **role** level, substantially enhances retrieval accuracy and efficiency. The **Role→Workflow** configuration provides the optimal balance between search breadth and specificity: role selection constrains the candidate space to a semantically coherent subset while preserving necessary workflow diversity. In this setup, the LLM-guided first layer functions as a high-level semantic filter, allowing the dense retriever to focus on fine-grained ranking. From a system design perspective, these results advocate for a scalable two-stage pipeline: first, utilizing a lightweight LLM to identify broad categories (e.g., roles), followed by focused dense retrieval within the targeted pool. This approach effectively handles large-scale workflow libraries without compromising precision.

6 Conclusion

In this paper, we presented FLOWSWITCH, an agentic framework that unifies state-aware transition detection with hierarchical retrieval to robustly manage workflow switching in multi-turn dialogues. Comprehensive experiments on FlowBench yield three critical insights: (1) hierarchical retrieval, particularly when anchored at the *role* level, offers the optimal trade-off between search breadth and precision; (2) self-generated search queries that explicitly articulate the target intent significantly outperform raw dialogue history as retrieval inputs; and (3) the workflow switch detector effectively acts as a gatekeeper, eliminating redundant search operations while boosting overall Top-k accuracy and MAP.

These results distill into actionable design principles for building scalable dialogue agents: (i) **Structure-First Retrieval:** Adopt multi-stage pipelines that leverage semantic categorization (e.g., roles) to narrow the search space before fine-grained ranking; (ii) **Query Refinement:** Prioritize the generation of semantically targeted search queries over using raw context; and (iii) **Explicit Control:** Integrate state-aware switching logic to minimize computational overhead and latency. Future work will explore adaptive hybrid retrieval mechanisms that dynamically weight dense and sparse signals, incorporate interactive clarification strategies for ambiguous user intents, and extend FLOWSWITCH to open-domain settings characterized by unstructured or evolving workflow repositories.

Acknowledgments

This work was financially supported by the National Science and Technology Council (NSTC) in Taiwan, under Grant 112-2223-E-002-012-MY5. We thank the National Center for High-performance Computing of National Institutes of Applied Research (NIAR) in Taiwan for providing computational and storage resources.

References

Manoj Ghuhan Arivazhagan, Lan Liu, Peng Qi, Xinchi Chen, William Yang Wang, and Zhiheng Huang. 2023. Hybrid hierarchical retrieval for open-domain question answering. In *Annual Meeting of the Association for Computational Linguistics*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6045–6049. IEEE.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757.

Akash Gupta, Ivaxi Sheth, Vyas Raina, Mark Gales, and Mario Fritz. 2024. LLM task interference: An initial study on the impact of task-switch in conversational history. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14633–14652, Miami, Florida, USA. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

O. Khattab and Matei A. Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Hui-Chi Kuo and Yun-Nung Chen. 2023. Zero-shot prompting for implicit intent prediction and recommendation with commonsense reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 249–258, Toronto, Canada. Association for Computational Linguistics.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *ArXiv*, abs/1609.01454.

Junhua Liu, Yong Keat Tan, Bin Fu, and Kwan Hui Lim. 2024. From intents to conversations: Generating intent-driven dialogues with contrastive learning for multi-turn classification.

Ye Liu, Kazuma Hashimoto, Yingbo Zhou, Semih Yavuz, Caiming Xiong, and Philip S. Yu. 2021. Dense hierarchical retrieval for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Representation Learning*, volume 2024, pages 9695–9717.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *AAAI Conference on Artificial Intelligence*.

Yuchen Shi, Siqi Cai, Zihan Xu, Yuei Qin, Gang Li, Hang Shao, Jiawei Chen, Deqing Yang, Ke Li, and Xing Sun. 2025. FlowAgent: Achieving Compliance and Flexibility for Workflow Agents. *arXiv preprint*. ArXiv:2502.14345 [cs].

Ming Sun, Yun-Nung Chen, and Alexander I Rudnicky. 2016a. HELPR: A framework to break the barrier across domains in spoken dialog systems. In *Dialogues with Social Robots: Enablements, Analyses, and Evaluation*, pages 257–269. Springer.

Ming Sun, Yun-Nung Chen, and Alexander I Rudnicky. 2016b. An intelligent assistant for high-level task understanding. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 169–174.

Xiaoyu Tan, Bin Li, Xihe Qiu, Chao Qu, Wei Chu, Yinghui Xu, and Yuan Qi. 2025. Meta-Agent-Workflow: Streamlining Tool Usage in LLMs through Workflow Construction, Retrieval, and Refinement. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, pages 458–467, New York, NY, USA. Association for Computing Machinery.

Hongru Wang, Wenyu Huang, Yang Deng, Rui Wang, Zezhong Wang, Yufei Wang, Fei Mi, Jeff Z. Pan, and Kam-Fai Wong. 2024. Unims-rag: A unified multi-source retrieval-augmented generation for personalized dialogue systems. *ArXiv*, abs/2401.13256.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Ruixuan Xiao, Wentao Ma, Ke Wang, Yuchuan Wu, Junbo Zhao, Haobo Wang, Fei Huang, and Yongbin Li. 2024. FlowBench: Revisiting and Benchmarking Workflow-Guided Planning for LLM-based Agents. *arXiv preprint*. ArXiv:2406.14884 [cs].

Lin Xu, Ningxin Peng, Daquan Zhou, See-Kiong Ng, and Jinlan Fu. 2024. Chain of thought explanation for dialogue state tracking. *ArXiv*, abs/2403.04656.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023.

ReAct: Synergizing reasoning and acting in language models. In *ICLR*. OpenReview.net.

Álvaro Zaera, Diana Nicoleta Popa, Ivan Sekulic, and Paolo Rosso. 2025. Efficient out-of-scope detection in dialogue systems via uncertainty-driven LLM routing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 328–335, Vienna, Austria. Association for Computational Linguistics.

Li-Ming Zhan, Haowen Liang, Bo Liu, Lu Fan, Xiao-Ming Wu, and Albert Y. S. Lam. 2021. Out-of-scope intent detection with self-supervision and discriminative training. In *Annual Meeting of the Association for Computational Linguistics*.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, et al. 2024. ProAgent: building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17591–17599.

A Hyperparameters

The hyperparameters used for `Qwen3-14B` are as follows:

- temperature: 0.7
- top-p: 0.95
- top-k: 20
- enable_thinking: True

For `Qwen3-8B`, the following settings are applied:

- temperature: 0.1
- enable_thinking: False

B Prompt Template

Domain Description Generation Prompt

```
### Instructions:
1. Write a high-level description of the provided domain based on the provided roles.
2. The description should be concise and clear without too many details.

### Domain:
{domain}
### Roles:
{roles}

return your summary in the key "summary" in json format
```

Role Description Generation Prompt

```
### Instructions:
1. Write a high-level description of the provided role based on the provided scenarios.
2. The description should be concise and clear without too many details.

### Role:
{role}
### Scenarios:
{scenarios}

return your summary in the key "summary" in json format
```

Workflow Summary Generation Prompt

```
### Instructions:
1. Write a high-level description of the provided workflow without too many details.

### Workflow Text:
{workflow_text}

return your summary in the key "summary" in json format
```

Workflow Switch Detector Prompt

```
### Instrctions:
Your goal is to help the user complete their task according to different workflow SOPs.
In order to accomplish this, you will need to understand the user's intention and determine the appropriate workflow SOPs to follow.
Specifically, given the current dialogue context and current workflow SOP, you will need to decide:
1. Whether the user's intention is aligned with the current workflow SOPs.
a. If the answer is no, based on your own knowledge, does current user's intention has to do with any potential tasks that might be described as a workflow SOP?
i. If yes, you will need generate a suitable search query to find the appropriate workflow SOP.
ii. If the answer is partially yes, you will need to search for any other workflow SOP that may be relevant to the user's intention.
iii. If no, you will need to stay with the current workflow, if the current context has nothing to do with any possible workflow SOPs and is out of current workflow's scope
b. If the answer is yes, you will need to stay with the current workflow.
2. Note that All you have to do is to decide which action to take, you do not need to take any other actions such as calling functions.
You only have 2 actions to choose from:
a. search: search for a suitable workflow SOP
b. stay: stay with the current workflow
3. It is possible that the current workflow SOPs is empty, then you will need to determine whether to search for suitable workflow SOPs or stay with the current workflow.
4. If you decide to search, the search query should be a clear and precise description of such workflow that can be used to tackle the user's intention, this should include the following information:
a. Potential Name of the workflow
b. Task description of the workflow
c. the action you need to take to solve the task
For example, the search query should be a string as follows:
```

```
"Potential Name of the workflow: Workflow Name, Task description of the workflow: Task Description, the action you need to take to solve the task: Action"
### Current Workflow SOPs:
{current_workflow_sop}
## Dialogue Context:
{dialogue_context}

### Output Format:
Follow the below format in every response under any circumstances:
“‘json
{{
"action": "<search, stay>",
"search_query": "<search query>",(empty if action is not search)
"user_intention": "<user intention>"
}}
“‘
### Response:
```

LLM as Retriever Prompt

```
Given the user query/dialogue history, please select the most relevant {selection_type}s from the candidates below.

User Query/Dialogue History:
{query}

Available {selection_type.capitalize()}s:
{candidate_text}

Please analyze the user’s intent and select the top {top_k} most relevant {selection_type}s that best match the user’s needs.
Return only the names of the selected {selection_type}s, one per line, in order of relevance (most relevant first).

Selected {selection_type}s
```

C Detailed Results

The breakdown results are detailed in Table 6, 7, 8, 9, and 10.

Retrieval Strategy	**LLM-Guided**	**Top-1**	**Top-2**	**Top-3**	**Top-4**	**Top-5**	**MAP**
Without Workflow Switch Detector							
Flat Retrieval	✗	41.1	54.1	59.5	63.0	65.4	52.7
Hierarchical (**D**→ **W**)	✗	30.8	42.2	46.8	49.8	51.6	40.9
	✓	36.4	48.5	54.3	58.1	60.7	46.6
Hierarchical (**D**→ **R**→ **W**)	✗	31.1	42.9	47.7	50.5	52.3	41.4
	✓	41.3	57.1	64.1	68.4	71.0	54.7
Hierarchical (**R**→ **W**)	✗	37.9	53.2	58.2	61.5	63.3	50.4
	✓	42.4	58.2	65.4	69.6	71.9	56.0
With Workflow Switch Detector							
Flat Retrieval	✗	63.8	69.3	72.0	73.8	75.0	77.0
Hierarchical (**D**→ **W**)	✗	61.2	66.3	68.8	70.3	71.3	73.0
	✓	62.8	67.9	70.6	72.5	73.7	75.2
Hierarchical (**D**→ **R** → **W**)	✗	61.4	66.8	69.4	70.8	71.8	73.4
	✓	65.0	72.4	76.1	78.7	80.3	80.0
Hierarchical (**R**→ **W**)	✗	62.6	70.0	72.8	74.8	76.1	76.1
	✓	65.4	72.9	76.9	79.4	80.9	80.7

Table 6: Workflow retrieval performance w/o & w/ workflow switch detector using BM25 in different settings (%).

Retriever	**Strategy**	**Query**	**LLM-Guided**	**Top-1**	**Top-2**	**Top-3**	**Top-4**	**Top-5**	**MAP**
BM25	Flat Retrieval	Full	✗	60.4	66.6	70.1	72.6	74.1	73.2
BM25	Flat Retrieval	Last3	✗	61.8	67.5	70.4	72.6	74.0	74.5
BM25	Flat Retrieval	Last2	✗	62.9	67.9	70.5	72.3	73.8	75.3
BM25	Flat Retrieval	Last1	✗	63.5	68.2	71.2	72.8	74.2	76.0
BM25	Flat Retrieval	Self-Gen	✗	67.9	71.4	73.0	73.6	74.0	80.5
BM25	Hierarchical (**D**→ **W**)	Full	✗	57.8	63.1	66.2	68.7	70.0	68.9
BM25	Hierarchical (**D**→ **W**)	Last3	✗	59.0	63.6	66.6	68.6	69.9	69.9
BM25	Hierarchical (**D**→ **W**)	Last2	✗	59.4	63.7	66.6	68.3	69.2	70.1
BM25	Hierarchical (**D**→ **W**)	Last1	✗	60.2	64.4	67.1	68.7	69.7	70.7
BM25	Hierarchical (**D**→ **W**)	Self-Gen	✗	65.9	69.6	70.9	71.4	71.9	77.8
BM25	Hierarchical (**D**→ **W**)	Full	✓	60.0	65.2	68.6	71.4	73.1	71.8
BM25	Hierarchical (**D**→ **W**)	Last3	✓	60.8	65.8	69.4	71.6	72.6	72.4
BM25	Hierarchical (**D**→ **W**)	Last2	✓	61.2	66.3	69.2	71.1	72.6	72.7
BM25	Hierarchical (**D**→ **W**)	Last1	✓	61.2	66.1	69.0	71.0	72.5	72.6
BM25	Hierarchical (**D**→ **W**)	Self-Gen	✓	64.8	68.4	70.1	70.9	71.8	76.1
BM25	Hierarchical (**R**→ **W**)	Full	✗	60.5	69.9	73.4	75.5	77.7	75.1
BM25	Hierarchical (**R**→ **W**)	Last3	✗	61.2	68.8	72.4	74.1	76.2	74.8
BM25	Hierarchical (**R**→ **W**)	Last2	✗	61.2	68.4	71.6	73.3	75.0	74.4
BM25	Hierarchical (**R**→ **W**)	Last1	✗	62.2	67.7	70.3	72.1	73.5	74.2
BM25	Hierarchical (**R**→ **W**)	Self-Gen	✗	65.4	68.8	70.1	71.3	71.7	76.9
BM25	Hierarchical (**R**→ **W**)	Full	✓	64.0	73.0	78.0	80.4	81.7	80.2
BM25	Hierarchical (**R**→ **W**)	Last3	✓	64.9	73.2	77.9	80.2	81.5	80.7
BM25	Hierarchical (**R**→ **W**)	Last2	✓	65.8	73.5	77.7	80.3	81.3	81.3
BM25	Hierarchical (**R**→ **W**)	Last1	✓	66.5	73.4	76.7	79.0	80.5	81.4
BM25	Hierarchical (**R**→ **W**)	Self-Gen	✓	65.4	71.9	74.4	78.0	79.8	79.8
BM25	Hierarchical (**D**→ **R** → **W**)	Full	✗	58.0	63.5	66.8	69.4	70.5	69.5
BM25	Hierarchical (**D**→ **R** → **W**)	Last3	✗	59.3	63.9	67.1	69.2	70.3	70.4
BM25	Hierarchical (**D**→ **R** → **W**)	Last2	✗	59.7	64.0	67.1	68.7	69.9	70.6
BM25	Hierarchical (**D**→ **R** → **W**)	Last1	✗	60.3	64.8	67.8	69.0	69.9	71.2
BM25	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✗	66.1	69.9	71.2	71.8	71.9	78.0
BM25	Hierarchical (**D**→ **R** → **W**)	Full	✓	64.2	72.5	77.0	79.6	81.0	79.9
BM25	Hierarchical (**D**→ **R** → **W**)	Last3	✓	64.8	73.1	76.7	79.1	80.7	80.3
BM25	Hierarchical (**D**→ **R** → **W**)	Last2	✓	65.6	73.1	76.9	79.2	80.5	80.7
BM25	Hierarchical (**D**→ **R** → **W**)	Last1	✓	66.0	72.8	76.0	78.3	79.6	80.7
BM25	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✓	64.8	71.6	74.0	76.3	78.3	79.0
E5	Flat Retrieval	Full	✗	62.1	71.5	75.0	76.9	78.4	77.5
E5	Flat Retrieval	Last3	✗	63.1	70.7	74.6	77.0	78.7	78.0
E5	Flat Retrieval	Last2	✗	63.8	71.8	75.6	77.4	78.8	78.8
E5	Flat Retrieval	Last1	✗	64.4	71.7	75.6	77.3	79.0	79.1
E5	Flat Retrieval	Self-Gen	✗	65.1	70.8	73.6	75.4	76.5	78.8
E5	Hierarchical (**D**→ **W**)	Full	✗	61.0	70.8	74.2	76.5	77.7	75.9
E5	Hierarchical (**D**→ **W**)	Last3	✗	62.0	69.9	73.8	75.9	77.1	76.1
E5	Hierarchical (**D**→ **W**)	Last2	✗	62.8	70.9	74.4	76.3	77.4	77.1
E5	Hierarchical (**D**→ **W**)	Last1	✗	63.4	71.1	74.4	76.2	77.6	77.3
E5	Hierarchical (**D**→ **W**)	Self-Gen	✗	63.3	69.0	72.1	74.0	75.1	76.2
E5	Hierarchical (**D**→ **W**)	Full	✓	63.9	73.2	76.9	78.4	79.6	79.4
E5	Hierarchical (**D**→ **W**)	Last3	✓	66.8	73.4	76.7	78.2	79.8	81.6
E5	Hierarchical (**D**→ **W**)	Last2	✓	67.7	74.4	77.7	79.0	80.3	82.8
E5	Hierarchical (**D**→ **W**)	Last1	✓	67.3	74.4	77.3	79.0	80.4	82.1
E5	Hierarchical (**D**→ **W**)	Self-Gen	✓	68.3	74.0	76.2	78.0	79.2	82.1
E5	Hierarchical (**R**→ **W**)	Full	✗	62.3	72.2	77.0	80.5	82.3	78.6
E5	Hierarchical (**R**→ **W**)	Last3	✗	63.2	71.6	77.2	80.3	81.6	78.9
E5	Hierarchical (**R**→ **W**)	Last2	✗	64.2	72.8	77.7	80.4	81.5	79.7
E5	Hierarchical (**R**→ **W**)	Last1	✗	65.2	72.8	77.2	79.8	81.1	80.1
E5	Hierarchical (**R**→ **W**)	Self-Gen	✗	65.7	70.8	73.4	75.4	77.4	78.3
E5	Hierarchical (**R**→ **W**)	Full	✓	64.4	75.1	80.9	82.8	83.6	81.5
E5	Hierarchical (**R**→ **W**)	Last3	✓	67.1	75.3	80.3	82.7	83.7	83.4
E5	Hierarchical (**R**→ **W**)	Last2	✓	67.7	76.1	80.6	82.6	83.3	83.9
E5	Hierarchical (**R**→ **W**)	Last1	✓	67.7	75.7	79.3	81.4	82.6	83.2
E5	Hierarchical (**R**→ **W**)	Self-Gen	✓	68.9	76.5	79.5	81.3	82.2	84.3
E5	Hierarchical (**D**→ **R** → **W**)	Full	✗	61.0	70.8	74.3	76.8	78.2	76.0
E5	Hierarchical (**D**→ **R** → **W**)	Last3	✗	62.0	69.8	73.7	76.1	77.4	76.1
E5	Hierarchical (**D**→ **R** → **W**)	Last2	✗	62.9	70.8	74.5	76.3	77.5	77.0
E5	Hierarchical (**D**→ **R** → **W**)	Last1	✗	63.5	70.9	74.4	76.3	77.5	77.3
E5	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✗	63.3	69.0	72.1	73.9	74.9	76.2
E5	Hierarchical (**D**→ **R** → **W**)	Full	✓	64.1	75.6	80.3	81.9	82.8	81.0
E5	Hierarchical (**D**→ **R** → **W**)	Last3	✓	67.1	76.3	79.6	81.8	82.7	83.1
E5	Hierarchical (**D**→ **R** → **W**)	Last2	✓	68.2	76.7	80.3	82.0	82.9	84.1
E5	Hierarchical (**D**→ **R** → **W**)	Last1	✓	67.9	76.2	79.0	81.0	82.1	83.2
E5	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✓	68.6	76.5	78.6	80.3	81.3	83.5

Table 7: Detailed workflow retrieval performance with FLOWSWITCH (Code)

Retriever	Strategy	Query	LLM-Guided	Top-1	Top-2	Top-3	Top-4	Top-5	MAP
BM25	Flat Retrieval	Full	✗	64.2	72.0	74.9	76.3	77.6	78.8
BM25	Flat Retrieval	Last3	✗	65.3	72.3	74.7	76.1	77.2	79.6
BM25	Flat Retrieval	Last2	✗	66.5	72.1	74.4	76.2	77.0	80.3
BM25	Flat Retrieval	Last1	✗	67.2	71.8	74.0	75.7	76.5	80.5
BM25	Flat Retrieval	Self-Gen	✗	68.4	72.6	74.5	75.2	75.5	81.6
BM25	Hierarchical (**D**→ **W**)	Full	✗	61.8	68.9	71.2	72.4	72.8	74.5
BM25	Hierarchical (**D**→ **W**)	Last3	✗	62.7	68.9	71.0	72.0	72.4	75.0
BM25	Hierarchical (**D**→ **W**)	Last2	✗	63.0	68.6	70.5	71.6	72.2	74.9
BM25	Hierarchical (**D**→ **W**)	Last1	✗	63.5	68.0	70.3	71.2	71.6	74.9
BM25	Hierarchical (**D**→ **W**)	Self-Gen	✗	66.8	71.1	72.6	73.2	73.5	79.2
BM25	Hierarchical (**D**→ **W**)	Full	✓	65.2	71.2	74.2	76.1	77.2	78.8
BM25	Hierarchical (**D**→ **W**)	Last3	✓	65.8	70.7	73.5	75.7	76.8	78.8
BM25	Hierarchical (**D**→ **W**)	Last2	✓	66.1	70.9	73.3	75.5	76.7	79.0
BM25	Hierarchical (**D**→ **W**)	Last1	✓	65.8	70.0	72.9	74.4	75.8	78.4
BM25	Hierarchical (**D**→ **W**)	Self-Gen	✓	66.9	72.4	74.4	75.5	76.3	80.2
BM25	Hierarchical (**R**→ **W**)	Full	✗	63.3	73.8	76.6	78.1	79.3	78.5
BM25	Hierarchical (**R**→ **W**)	Last3	✗	63.8	72.8	75.4	76.7	77.9	78.0
BM25	Hierarchical (**R**→ **W**)	Last2	✗	63.9	71.9	74.2	75.6	76.7	77.5
BM25	Hierarchical (**R**→ **W**)	Last1	✗	64.1	69.7	72.0	73.6	74.8	76.3
BM25	Hierarchical (**R**→ **W**)	Self-Gen	✗	66.2	70.9	72.6	73.9	75.1	78.3
BM25	Hierarchical (**R**→ **W**)	Full	✓	66.3	75.4	79.6	81.6	82.5	82.6
BM25	Hierarchical (**R**→ **W**)	Last3	✓	67.6	75.4	79.6	81.2	82.2	83.2
BM25	Hierarchical (**R**→ **W**)	Last2	✓	68.1	75.3	79.2	81.1	82.0	83.4
BM25	Hierarchical (**R**→ **W**)	Last1	✓	67.4	74.6	78.5	80.2	81.0	82.6
BM25	Hierarchical (**R**→ **W**)	Self-Gen	✓	67.3	74.8	78.3	79.7	80.9	82.3
BM25	Hierarchical (**D**→ **R** → **W**)	Full	✗	61.8	69.3	71.8	72.5	73.1	74.6
BM25	Hierarchical (**D**→ **R** → **W**)	Last3	✗	62.7	69.2	71.6	72.1	72.7	75.1
BM25	Hierarchical (**D**→ **R** → **W**)	Last2	✗	63.0	68.8	70.8	71.5	72.3	74.9
BM25	Hierarchical (**D**→ **R** → **W**)	Last1	✗	63.4	68.0	70.4	71.0	72.0	74.8
BM25	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✗	67.0	71.2	72.7	73.3	73.6	79.4
BM25	Hierarchical (**D**→ **R** → **W**)	Full	✓	65.9	74.8	78.8	80.7	81.7	81.8
BM25	Hierarchical (**D**→ **R** → **W**)	Last3	✓	67.3	75.1	78.5	80.3	81.3	82.6
BM25	Hierarchical (**D**→ **R** → **W**)	Last2	✓	67.6	74.6	78.3	80.5	81.3	82.7
BM25	Hierarchical (**D**→ **R** → **W**)	Last1	✓	67.1	74.1	77.5	79.4	80.4	82.0
BM25	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✓	66.9	74.5	77.6	79.1	79.9	81.6
E5	Flat Retrieval	Full	✗	63.9	72.2	75.5	77.3	78.5	78.7
E5	Flat Retrieval	Last3	✗	64.7	71.5	74.9	77.2	78.8	79.0
E5	Flat Retrieval	Last2	✗	65.2	72.6	76.3	77.8	78.8	79.8
E5	Flat Retrieval	Last1	✗	65.8	72.6	76.0	77.2	78.7	80.0
E5	Flat Retrieval	Self-Gen	✗	66.1	72.1	74.9	76.6	77.2	80.0
E5	Hierarchical (**D**→ **W**)	Full	✗	62.7	71.8	74.5	76.6	77.6	77.1
E5	Hierarchical (**D**→ **W**)	Last3	✗	63.5	70.8	74.3	76.2	77.2	77.3
E5	Hierarchical (**D**→ **W**)	Last2	✗	64.2	72.2	75.2	76.6	77.5	78.2
E5	Hierarchical (**D**→ **W**)	Last1	✗	64.8	72.1	75.0	76.3	77.6	78.4
E5	Hierarchical (**D**→ **W**)	Self-Gen	✗	64.7	70.8	73.5	75.3	76.0	77.9
E5	Hierarchical (**D**→ **W**)	Full	✓	63.5	74.0	77.0	78.7	79.8	79.2
E5	Hierarchical (**D**→ **W**)	Last3	✓	66.7	74.0	77.1	78.6	79.8	81.4
E5	Hierarchical (**D**→ **W**)	Last2	✓	67.7	74.7	77.7	78.9	79.8	82.6
E5	Hierarchical (**D**→ **W**)	Last1	✓	68.4	75.2	77.8	79.0	80.1	82.7
E5	Hierarchical (**D**→ **W**)	Self-Gen	✓	68.5	73.1	75.3	76.7	77.7	81.4
E5	Hierarchical (**R**→ **W**)	Full	✗	64.1	72.8	77.2	80.2	81.9	79.7
E5	Hierarchical (**R**→ **W**)	Last3	✗	64.7	72.3	77.5	80.1	81.2	79.8
E5	Hierarchical (**R**→ **W**)	Last2	✗	65.6	73.5	78.1	80.1	81.1	80.6
E5	Hierarchical (**R**→ **W**)	Last1	✗	66.5	73.5	77.3	79.5	80.8	81.0
E5	Hierarchical (**R**→ **W**)	Self-Gen	✗	66.5	72.2	74.8	76.3	77.7	79.3
E5	Hierarchical (**R**→ **W**)	Full	✓	63.9	76.4	80.7	82.3	82.6	81.0
E5	Hierarchical (**R**→ **W**)	Last3	✓	67.1	77.2	80.0	82.0	82.5	83.2
E5	Hierarchical (**R**→ **W**)	Last2	✓	68.0	77.2	80.0	82.1	82.4	83.9
E5	Hierarchical (**R**→ **W**)	Last1	✓	68.9	77.1	79.7	81.5	81.9	84.0
E5	Hierarchical (**R**→ **W**)	Self-Gen	✓	69.5	76.2	79.2	80.7	81.4	84.0
E5	Hierarchical (**D**→ **R** → **W**)	Full	✗	62.7	71.8	74.7	76.9	78.1	77.1
E5	Hierarchical (**D**→ **R** → **W**)	Last3	✗	63.6	70.8	74.3	76.4	77.5	77.3
E5	Hierarchical (**D**→ **R** → **W**)	Last2	✗	64.4	72.1	75.2	76.7	77.5	78.2
E5	Hierarchical (**D**→ **R** → **W**)	Last1	✗	65.0	72.0	74.9	76.4	77.5	78.5
E5	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✗	64.8	70.8	73.5	74.9	75.9	77.9
E5	Hierarchical (**D**→ **R** → **W**)	Full	✓	63.7	76.0	79.5	81.3	82.0	80.5
E5	Hierarchical (**D**→ **R** → **W**)	Last3	✓	66.8	76.7	79.5	81.2	81.6	82.6
E5	Hierarchical (**D**→ **R** → **W**)	Last2	✓	68.0	77.2	79.7	81.3	81.8	83.6
E5	Hierarchical (**D**→ **R** → **W**)	Last1	✓	68.6	76.7	79.0	80.6	81.3	83.4
E5	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✓	69.1	75.7	78.4	79.5	80.4	83.1

Table 8: Detailed workflow retrieval performance with FLOWSWITCH (Flowchart)

Retriever	Strategy	Query	LLM-Guided	Top-1	Top-2	Top-3	Top-4	Top-5	MAP
BM25	Flat Retrieval	Full	✗	61.2	67.9	71.5	73.7	75.3	74.6
BM25	Flat Retrieval	Last3	✗	62.2	68.1	71.2	73.4	74.7	75.1
BM25	Flat Retrieval	Last2	✗	63.0	68.6	71.2	73.2	74.5	75.7
BM25	Flat Retrieval	Last1	✗	63.5	67.8	70.4	72.6	73.9	75.5
BM25	Flat Retrieval	Self-Gen	✗	66.8	70.7	72.8	74.0	74.9	79.7
BM25	Hierarchical (**D**→ **W**)	Full	✗	59.2	65.8	68.5	70.8	71.9	71.3
BM25	Hierarchical (**D**→ **W**)	Last3	✗	59.9	65.6	68.3	70.3	71.8	71.5
BM25	Hierarchical (**D**→ **W**)	Last2	✗	60.3	65.3	68.0	70.2	71.2	71.6
BM25	Hierarchical (**D**→ **W**)	Last1	✗	60.6	64.9	67.7	69.7	70.5	71.6
BM25	Hierarchical (**D**→ **W**)	Self-Gen	✗	65.4	69.1	70.9	71.9	73.0	77.6
BM25	Hierarchical (**D**→ **W**)	Full	✓	59.5	64.7	67.5	69.5	71.2	71.0
BM25	Hierarchical (**D**→ **W**)	Last3	✓	60.5	64.7	67.3	69.2	70.6	71.4
BM25	Hierarchical (**D**→ **W**)	Last2	✓	61.0	64.9	67.6	69.3	70.9	71.7
BM25	Hierarchical (**D**→ **W**)	Last1	✓	60.1	64.2	66.7	68.4	70.0	70.9
BM25	Hierarchical (**D**→ **W**)	Self-Gen	✓	62.5	65.6	67.5	69.6	71.2	73.8
BM25	Hierarchical (**R**→ **W**)	Full	✗	60.7	71.0	74.3	77.9	78.6	75.5
BM25	Hierarchical (**R**→ **W**)	Last3	✗	61.4	70.3	73.5	76.4	77.3	75.3
BM25	Hierarchical (**R**→ **W**)	Last2	✗	62.3	69.6	72.8	75.6	76.6	75.4
BM25	Hierarchical (**R**→ **W**)	Last1	✗	62.6	68.6	71.8	73.7	74.7	74.7
BM25	Hierarchical (**R**→ **W**)	Self-Gen	✗	65.4	69.8	71.8	73.1	74.2	77.5
BM25	Hierarchical (**R**→ **W**)	Full	✓	62.2	70.1	74.4	78.0	80.4	77.1
BM25	Hierarchical (**R**→ **W**)	Last3	✓	63.8	69.9	74.8	77.9	79.9	77.9
BM25	Hierarchical (**R**→ **W**)	Last2	✓	64.3	70.2	74.5	77.7	80.1	78.4
BM25	Hierarchical (**R**→ **W**)	Last1	✓	63.9	69.5	73.4	76.7	79.2	77.8
BM25	Hierarchical (**R**→ **W**)	Self-Gen	✓	64.5	70.1	73.8	77.2	79.0	78.2
BM25	Hierarchical (**D**→ **R** → **W**)	Full	✗	59.3	66.7	69.7	72.0	73.3	71.9
BM25	Hierarchical (**D**→ **R** → **W**)	Last3	✗	60.0	66.4	69.4	71.7	72.6	72.0
BM25	Hierarchical (**D**→ **R** → **W**)	Last2	✗	60.4	66.2	69.0	71.3	72.0	72.1
BM25	Hierarchical (**D**→ **R** → **W**)	Last1	✗	60.8	65.5	68.4	70.2	71.2	71.9
BM25	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✗	65.6	69.3	71.1	72.5	73.4	77.9
BM25	Hierarchical (**D**→ **R** → **W**)	Full	✓	61.8	69.8	74.2	77.8	80.3	76.6
BM25	Hierarchical (**D**→ **R** → **W**)	Last3	✓	63.2	69.4	73.8	77.4	79.7	77.3
BM25	Hierarchical (**D**→ **R** → **W**)	Last2	✓	63.9	69.3	73.6	77.4	80.3	78.0
BM25	Hierarchical (**D**→ **R** → **W**)	Last1	✓	63.5	68.8	72.9	76.3	78.7	77.2
BM25	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✓	63.7	69.4	73.0	76.5	78.5	77.5
E5	Flat Retrieval	Full	✗	62.5	71.4	74.8	76.8	78.1	77.3
E5	Flat Retrieval	Last3	✗	63.4	71.2	74.7	76.9	78.4	77.9
E5	Flat Retrieval	Last2	✗	63.9	72.0	75.9	77.5	78.6	78.7
E5	Flat Retrieval	Last1	✗	64.6	71.8	75.8	77.2	78.5	79.0
E5	Flat Retrieval	Self-Gen	✗	64.9	70.5	73.5	75.0	76.5	78.5
E5	Hierarchical (**D**→ **W**)	Full	✗	61.5	71.2	74.0	76.3	77.3	76.0
E5	Hierarchical (**D**→ **W**)	Last3	✗	62.3	70.5	73.9	75.8	76.7	76.2
E5	Hierarchical (**D**→ **W**)	Last2	✗	62.9	71.4	74.8	76.4	77.2	77.1
E5	Hierarchical (**D**→ **W**)	Last1	✗	63.5	71.3	74.5	75.9	77.2	77.2
E5	Hierarchical (**D**→ **W**)	Self-Gen	✗	63.4	69.0	72.1	74.1	75.3	76.4
E5	Hierarchical (**D**→ **W**)	Full	✓	61.2	67.9	71.7	73.0	74.7	73.4
E5	Hierarchical (**D**→ **W**)	Last3	✓	62.3	68.4	72.1	74.2	76.0	74.9
E5	Hierarchical (**D**→ **W**)	Last2	✓	62.8	69.6	72.9	75.2	76.8	75.6
E5	Hierarchical (**D**→ **W**)	Last1	✓	64.0	70.0	73.3	75.3	76.7	76.8
E5	Hierarchical (**D**→ **W**)	Self-Gen	✓	64.0	68.5	70.9	72.5	73.7	74.6
E5	Hierarchical (**R**→ **W**)	Full	✗	62.7	72.4	76.7	80.2	81.7	78.5
E5	Hierarchical (**R**→ **W**)	Last3	✗	63.4	72.1	77.3	79.9	81.1	78.8
E5	Hierarchical (**R**→ **W**)	Last2	✗	64.2	72.9	77.9	79.8	80.8	79.4
E5	Hierarchical (**R**→ **W**)	Last1	✗	65.1	73.0	77.1	79.3	80.4	79.7
E5	Hierarchical (**R**→ **W**)	Self-Gen	✗	65.5	70.7	73.7	75.1	77.2	78.0
E5	Hierarchical (**R**→ **W**)	Full	✓	62.3	73.5	77.8	81.2	82.2	78.3
E5	Hierarchical (**R**→ **W**)	Last3	✓	64.2	74.5	78.2	81.2	82.0	79.9
E5	Hierarchical (**R**→ **W**)	Last2	✓	65.1	74.7	78.7	81.0	81.6	80.4
E5	Hierarchical (**R**→ **W**)	Last1	✓	66.0	74.5	78.5	80.8	81.7	80.9
E5	Hierarchical (**R**→ **W**)	Self-Gen	✓	65.8	72.1	74.6	77.0	79.3	78.6
E5	Hierarchical (**D**→ **R** → **W**)	Full	✗	61.6	71.2	74.1	76.7	77.8	76.1
E5	Hierarchical (**D**→ **R** → **W**)	Last3	✗	62.3	70.5	74.0	76.1	77.1	76.2
E5	Hierarchical (**D**→ **R** → **W**)	Last2	✗	63.0	71.2	74.8	76.4	77.3	77.1
E5	Hierarchical (**D**→ **R** → **W**)	Last1	✗	63.7	71.2	74.4	76.0	77.1	77.2
E5	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✗	63.8	69.2	72.1	74.0	75.3	76.6
E5	Hierarchical (**D**→ **R** → **W**)	Full	✓	62.1	72.3	76.3	79.2	81.3	77.5
E5	Hierarchical (**D**→ **R** → **W**)	Last3	✓	63.8	73.1	76.9	79.8	81.2	78.9
E5	Hierarchical (**D**→ **R** → **W**)	Last2	✓	64.9	74.1	77.8	79.9	81.1	79.8
E5	Hierarchical (**D**→ **R** → **W**)	Last1	✓	65.6	74.1	77.3	79.6	80.9	80.2
E5	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✓	65.4	70.6	73.5	76.0	78.2	77.7

Table 9: Detailed workflow retrieval performance with FlowSwitch (Summary)

Retriever	Strategy	Query	LLM-Guided	Top-1	Top-2	Top-3	Top-4	Top-5	MAP
BM25	Flat Retrieval	Full	✗	59.9	67.1	71.2	73.2	74.9	74.4
BM25	Flat Retrieval	Last3	✗	61.3	67.7	70.9	73.1	74.5	75.3
BM25	Flat Retrieval	Last2	✗	62.6	67.9	71.1	73.0	74.5	76.0
BM25	Flat Retrieval	Last1	✗	62.7	68.3	71.1	72.6	74.0	76.0
BM25	Flat Retrieval	Self-Gen	✗	63.8	69.1	71.4	72.9	74.2	77.7
BM25	Hierarchical (**D**→ **W**)	Full	✗	57.9	64.7	67.8	69.6	71.3	70.9
BM25	Hierarchical (**D**→ **W**)	Last3	✗	59.1	64.6	67.8	69.4	70.7	71.4
BM25	Hierarchical (**D**→ **W**)	Last2	✗	59.4	64.5	67.5	69.0	70.5	71.4
BM25	Hierarchical (**D**→ **W**)	Last1	✗	60.0	64.5	66.9	68.5	69.7	71.4
BM25	Hierarchical (**D**→ **W**)	Self-Gen	✗	62.1	67.3	69.9	70.9	72.1	75.4
BM25	Hierarchical (**D**→ **W**)	Full	✓	61.6	69.9	73.5	75.5	76.8	76.8
BM25	Hierarchical (**D**→ **W**)	Last3	✓	63.5	70.3	73.1	74.8	75.8	77.6
BM25	Hierarchical (**D**→ **W**)	Last2	✓	63.7	69.9	72.7	74.4	75.6	77.4
BM25	Hierarchical (**D**→ **W**)	Last1	✓	62.6	68.1	71.0	73.1	74.0	75.6
BM25	Hierarchical (**D**→ **W**)	Self-Gen	✓	62.8	68.1	70.5	72.1	73.1	76.1
BM25	Hierarchical (**R**→ **W**)	Full	✗	59.9	71.2	74.8	77.5	78.7	75.9
BM25	Hierarchical (**R**→ **W**)	Last3	✗	61.1	70.2	73.4	76.0	77.4	75.9
BM25	Hierarchical (**R**→ **W**)	Last2	✗	61.7	69.8	72.7	75.1	76.5	75.8
BM25	Hierarchical (**R**→ **W**)	Last1	✗	61.5	68.1	71.0	72.7	74.0	74.4
BM25	Hierarchical (**R**→ **W**)	Self-Gen	✗	63.8	69.4	72.2	73.7	75.3	77.0
BM25	Hierarchical (**R**→ **W**)	Full	✓	63.9	74.4	78.8	80.9	81.7	81.1
BM25	Hierarchical (**R**→ **W**)	Last3	✓	65.7	73.9	78.0	80.3	81.6	81.9
BM25	Hierarchical (**R**→ **W**)	Last2	✓	66.6	74.2	77.9	80.6	81.6	82.5
BM25	Hierarchical (**R**→ **W**)	Last1	✓	65.5	72.7	76.8	79.4	80.6	81.2
BM25	Hierarchical (**R**→ **W**)	Self-Gen	✓	65.1	72.0	76.3	78.6	80.1	80.5
BM25	Hierarchical (**D**→ **R** → **W**)	Full	✗	58.1	65.6	68.8	70.3	71.9	71.4
BM25	Hierarchical (**D**→ **R** → **W**)	Last3	✗	59.3	65.4	68.6	70.0	71.1	71.8
BM25	Hierarchical (**D**→ **R** → **W**)	Last2	✗	59.6	65.5	68.4	69.6	70.8	71.8
BM25	Hierarchical (**D**→ **R** → **W**)	Last1	✗	60.3	65.3	67.4	69.1	70.3	71.9
BM25	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✗	62.6	68.0	70.3	71.2	72.5	75.9
BM25	Hierarchical (**D**→ **R** → **W**)	Full	✓	63.0	73.7	77.9	80.2	81.6	80.1
BM25	Hierarchical (**D**→ **R** → **W**)	Last3	✓	65.2	73.2	77.3	79.8	81.3	81.2
BM25	Hierarchical (**D**→ **R** → **W**)	Last2	✓	65.7	73.6	77.1	79.5	81.0	81.5
BM25	Hierarchical (**D**→ **R** → **W**)	Last1	✓	64.7	72.5	76.3	78.7	80.4	80.4
BM25	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✓	64.5	71.3	75.2	77.8	79.6	79.6
E5	Flat Retrieval	Full	✗	61.0	69.9	73.4	75.5	77.1	76.6
E5	Flat Retrieval	Last3	✗	62.2	70.3	73.8	75.9	77.5	77.5
E5	Flat Retrieval	Last2	✗	62.3	71.2	74.9	76.7	77.8	78.1
E5	Flat Retrieval	Last1	✗	63.5	71.2	74.7	76.2	77.6	78.5
E5	Flat Retrieval	Self-Gen	✗	63.9	69.8	72.7	74.4	75.7	78.4
E5	Hierarchical (**D**→ **W**)	Full	✗	60.0	69.8	73.1	75.7	76.7	75.4
E5	Hierarchical (**D**→ **W**)	Last3	✗	60.9	69.5	73.0	75.0	76.2	75.8
E5	Hierarchical (**D**→ **W**)	Last2	✗	61.3	70.5	74.0	75.6	76.6	76.5
E5	Hierarchical (**D**→ **W**)	Last1	✗	62.4	70.4	73.6	75.3	76.6	76.9
E5	Hierarchical (**D**→ **W**)	Self-Gen	✗	62.1	68.1	71.1	73.1	74.3	76.0
E5	Hierarchical (**D**→ **W**)	Full	✓	60.5	67.4	70.6	73.3	75.5	74.1
E5	Hierarchical (**D**→ **W**)	Last3	✓	62.5	69.0	72.2	74.1	75.5	76.3
E5	Hierarchical (**D**→ **W**)	Last2	✓	62.4	69.6	73.7	75.3	77.0	76.9
E5	Hierarchical (**D**→ **W**)	Last1	✓	63.5	70.5	73.7	75.5	76.8	77.6
E5	Hierarchical (**D**→ **W**)	Self-Gen	✓	64.4	70.6	72.3	74.2	75.7	78.3
E5	Hierarchical (**R**→ **W**)	Full	✗	61.2	71.1	75.8	79.4	81.1	77.9
E5	Hierarchical (**R**→ **W**)	Last3	✗	62.3	71.3	76.2	79.0	80.3	78.4
E5	Hierarchical (**R**→ **W**)	Last2	✗	62.6	72.2	77.0	79.2	80.3	78.8
E5	Hierarchical (**R**→ **W**)	Last1	✗	64.1	72.2	76.2	78.8	79.9	79.4
E5	Hierarchical (**R**→ **W**)	Self-Gen	✗	64.2	70.1	73.0	74.7	76.6	77.5
E5	Hierarchical (**R**→ **W**)	Full	✓	60.6	72.0	78.7	81.3	82.1	77.9
E5	Hierarchical (**R**→ **W**)	Last3	✓	62.6	72.7	78.4	80.8	81.6	79.3
E5	Hierarchical (**R**→ **W**)	Last2	✓	62.8	73.0	78.6	80.5	81.5	79.5
E5	Hierarchical (**R**→ **W**)	Last1	✓	63.5	73.4	78.0	79.9	81.0	79.6
E5	Hierarchical (**R**→ **W**)	Self-Gen	✓	65.9	72.4	76.8	79.1	80.2	81.0
E5	Hierarchical (**D**→ **R** → **W**)	Full	✗	60.0	69.8	73.3	76.0	77.2	75.4
E5	Hierarchical (**D**→ **R** → **W**)	Last3	✗	61.0	69.4	73.0	75.2	76.5	75.8
E5	Hierarchical (**D**→ **R** → **W**)	Last2	✗	61.4	70.4	74.0	75.6	76.6	76.5
E5	Hierarchical (**D**→ **R** → **W**)	Last1	✗	62.6	70.3	73.5	75.5	76.6	76.9
E5	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✗	62.3	68.2	71.4	73.2	74.4	76.0
E5	Hierarchical (**D**→ **R** → **W**)	Full	✓	60.8	71.4	77.8	80.2	81.3	77.5
E5	Hierarchical (**D**→ **R** → **W**)	Last3	✓	63.1	72.6	77.7	80.0	80.8	79.2
E5	Hierarchical (**D**→ **R** → **W**)	Last2	✓	63.3	73.7	78.1	79.8	80.8	79.6
E5	Hierarchical (**D**→ **R** → **W**)	Last1	✓	64.3	74.1	77.6	79.1	80.4	80.1
E5	Hierarchical (**D**→ **R** → **W**)	Self-Gen	✓	65.4	71.8	75.8	78.5	79.8	80.2

Table 10: Detailed workflow retrieval performance with FlowSwitch (Text)