Task-Oriented Dialogue Systems through Function Calling

Tiziano Labruna
University of Bozen-Bolzano
Fondazione Bruno Kessler
tlabruna@fbk.eu

Giovanni Bonetta
Fondazione Bruno Kessler
gbonetta@fbk.eu

Bernardo Magnini
Fondazione Bruno Kessler
magnini@fbk.eu

Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities in generating dialogues and handling a broad range of user queries. However, their effectiveness as end-to-end Task-Oriented Dialogue (TOD) systems remains limited due to their reliance on static parametric memory, which fails to accommodate evolving knowledge bases (KBs). This paper investigates a scalable function-calling approach that enables LLMs to retrieve only the necessary KB entries via schema-guided queries, rather than embedding the entire KB into each prompt. This selective retrieval strategy reduces prompt size and inference time while improving factual accuracy in system responses. We evaluate our method on the MultiWOZ 2.3 dataset and compare it against a full-KB baseline that injects the entire KB into every prompt. Experimental results show that our approach consistently outperforms the full-KB method in accuracy, while requiring significantly fewer input tokens and considerably less computation time, especially when the KB size increases.

1 Introduction

Large Language Models (LLMs) have recently achieved human-level fluency on a wide range of language tasks, thanks to scaling laws and instruction tuning (Brown et al., 2020; Wei et al., 2022). Among the many applications of LLMs, task-oriented dialogue (TOD) systems are designed to assist users in completing specific goals, such as booking a flight, ordering food, or finding information, through natural language interactions. Despite their fluency, LLMs face a fundamental limitation in TOD scenarios: the knowledge encoded in their parametric memory is frozen at training time, preventing the model from reflecting subsequent changes in the external world.

A straightforward workaround is to embed the *entire* KB in each prompt, ensuring that every conversational turn has access to all facts that might be needed. Unfortunately, KBs quickly outgrow the context window and contain much information that is irrelevant to the current user request. Padding the prompt with thousands of unused entries inflates latency and inference cost while making it harder for the decoder to focus on the truly salient rows.

Retrieval-augmented generation (RAG) (Lewis et al., 2020) mitigates prompt bloat for *unstructured* corpora by retrieving the top-k textual chunks at inference time. However, MultiWOZ-style KBs are json objects that closely resembles relational-rows, which often differ by only one or two cells (e.g. the name of two cafés on the same street). Converting each row into an ad-hoc sentence has been shown possible (Roberti et al., 2020), but could have limitations such us: it destroys schema information, it blows up the index with near-duplicate passages, and it confuses dense retrievers that rely on lexical diversity.

Recent variants such as TableRAG (Chen et al., 2024), SRAG (Lin et al., 2025) and the survey of Soliman et al. (Soliman and Gurevych, 2025) confirm that naïve, chunk-based RAG loses effectiveness on highly structured or repetitive data. Rather than engineering a specialised, table-aware retriever, we exploit the KB's schema directly via *function calls*: slot values detected in the user utterance are mapped to structured query parameters, yielding constant-time access to exactly those rows that the answer requires.

Building on recent work that enables LLMs to invoke external tools—e.g. ReAct (Yao et al.), Toolformer (Schick et al., 2023), and ToolLLM (Qin et al., 2023)—we introduce a selective function-calling pipeline for TOD. At each turn the model

Proceedings of Recent Advances in Natural Language Processing,pages 614–622
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-072

(i) detects whether the user's intent contains slot values, (ii) issues a targeted query against the KB, and (iii) integrates only the returned rows into the prompt. Evaluated on MultiWOZ 2.3 (Zang et al., 2020), this strategy cuts prompt tokens by ~ 4 times while improving factual accuracy over a baseline that serialises the full KB at every turn.

2 Related Work

TOD systems traditionally rely on modular architectures that separate natural language understanding, dialogue state tracking, and policy learning components. However, recent advances in LLMs have motivated a shift toward end-to-end TOD systems that aim to unify these components within a single generative model (Zhao et al., 2023). Despite their fluency, LLMs struggle to maintain factual accuracy and adapt to dynamic knowledge bases, due to their reliance on static parametric memory.

To address the rigidity of LLMs in dynamic settings, retrieval-augmented generation (RAG) methods have emerged as a promising solution. These methods combine LLMs with external retrievers that provide relevant context at inference time (Izacard et al., 2023; Zhang et al., 2022). Nonetheless, RAG was designed for free-text collections and shows degradation when the source is highly structured.

A recent survey on RAG over tabular data (Soliman and Gurevych, 2025) highlights three open issues: schema loss during chunking, retrieval ambiguity caused by repetitive content, and inflated context windows when multiple near-identical KB samples are returned. Our work departs from RAG entirely: by delegating KB access to an external function, we preserve the structure of the data, keep inference costs predictable, avoid embedding redundancy and its inherent risk of false positive retrieval, since the function calling logic is simple rule-based python code.

A recent trend involves leveraging function calling APIs to handle structured queries, allowing LLMs to delegate knowledge access to external tools (Yao et al.; Qin et al., 2023). These approaches improve both factual grounding and system modularity by decoupling reasoning from knowledge access. Similar strategies have been employed in dynamic dialogue settings to mitigate prompt length limitations and reduce inference cost (Mialon et al., 2023).

Compared to approaches that embed the entire KB in the prompt, our method adopts a selective function-calling strategy triggered by slot-value detection. This allows for minimal and relevant KB access, reducing computational overhead and improving response relevance. Our work builds upon these recent advances by offering a practical and scalable framework for integrating dynamic retrieval into end-to-end TOD systems.

Recent work by Labruna et al. (2024) tackles a related challenge in the question answering domain: determining whether an LLM's parametric knowledge is sufficient to answer a query, or whether external retrieval is required. While their task differs in format and scope, the core objective aligns with ours: making informed, context-dependent decisions about when to rely on internal memory versus external knowledge. We extend this idea to multi-turn, structured dialogue by integrating schema-guided function calls instead of free-text retrieval.

3 Function Calling Methodology

In this study, we propose a TOD system designed to selectively retrieve information from a knowledge base when necessary, based on the specific needs expressed in each user turn. The system is built around a large language model that manages the conversation and executes knowledge retrieval when appropriate.

It begins with a system prompt that sets the initial behavioral instructions for the LLM (Section 3.1). Then each user turn is analyzed to determine whether external knowledge is required. This is achieved through a lightweight slot-value and intent extraction process (Section 3.2). For instance, a vague input like "Can you help me?" does not lead to a KB query, while a more specific request such as "Find me a cheap Italian restaurant in the center" does.

When relevant information is detected, it is passed to a function that maps the extracted content into structured query parameters (Section 3.3). These parameters are then used to retrieve only the most relevant KB entries (Section 3.4). The results of the query are evaluated for adequacy and informativeness (Section 3.5), and, if appropriate, integrated into the LLM's prompt to inform the response. This targeted augmentation ensures that the model's outputs are both accurate and aligned with the user's

intent.

To support longer conversations, the system includes a conversation management component that trims older dialogue turns as needed to remain within the LLM's context window, while preserving relevant history (Section 3.6).

Through this structured and dynamic methodology, the system is able to maintain a high level of relevance, responsiveness, and efficiency in handling complex task-oriented dialogues.

3.1 Initial System Prompt

The core LLM at the basis of our dialogue system is guided through an initial system prompt. This prompt defines the model's general role in the conversation, for instance, acting as a travel assistant or a booking agent, and provides domain-specific instructions that shape its behavior throughout the interaction. It may also include stylistic directives to ensure consistent tone and phrasing across responses. For example:

```
You are an airline ticket booking
system.  Respond politely and
provide suggestions for available
flights.
```

This initial prompt establishes the context and objectives of the conversation, enabling the model to generate coherent and purpose-aligned responses as the dialogue evolves.

3.2 Slot-Value and Intent Extraction

Once the user submits an input, the system (LLM) receives it and begins the process of understanding the user's request. To enable targeted retrieval of knowledge, we first identify whether the input contains any slot-values. This is done by forwarding the user input to a second language model specifically tasked with slot-value extraction.

This model is provided with a predefined list of possible slot types relevant to the current domain, each accompanied by a brief description. However, the list of possible values for each slot is not given, as these must be inferred by the model using its parametric knowledge.

If the model returns that no slot-values have been detected, the initial LLM proceeds to generate a response using only the context available in the current conversation history.

If slot-values are found, the system proceeds to determine the user's intent. This is achieved by prompting another model (or another instance of the same model) with the same user input and a list of possible intents, each with a short description. The objective is to infer the communicative goal behind the user's utterance.

With both slot-values and intent identified, the system is equipped to determine whether external knowledge retrieval is needed and how to formulate a precise query for it.

3.3 Function Instruction and Parameter Mapping

After extracting slot-values and the user's intent, the system determines whether a query to an external KB is necessary. If so, these extracted elements are used to select the most suitable function from a predefined set, where each function f_i corresponds to a specific type of query designed to retrieve domain-relevant information.

Each function is defined with a set of parameters $\theta_1, \theta_2, \ldots, \theta_n$ that correspond to expected slot-values. Formally, we represent a function as:

$$f\_i(\theta\_1, \theta\_2, \ldots, \theta\_n) \quad (1)$$

For example, a function to search for restaurants based on area, price range, and cuisine may be defined as:

Function: SearchRestaurant(Area, PriceRange, Cuisine)

When the slot-value extraction step identifies relevant inputs (for instance, "northern" as a value for the slot *Area*) and the intent is classified as *search*, these are passed to the function selection logic, which results in the following function call:

$$f\_\text{SearchRestaurant}(\theta\_\text{Area} = \text{"north"}) \quad (2)$$

Only the slots that were detected in the input are passed as arguments to the selected function, while missing parameters remain unspecified. This targeted mapping enables the system to retrieve only the KB entries necessary to respond to the user's

request, avoiding redundant data access and improving both efficiency and relevance.

3.4 Function Call and Query Execution

After selecting the appropriate function f_i and assigning its parameters based on the extracted slot-values, the system proceeds to execute the function to query the knowledge base (KB). Each function is tailored to a specific domain and is designed to filter KB entries according to the input constraints provided in its parameters.

The output of the function call is a set of retrieved instances:

$$\mathcal{I} = i\_1, i\_2, \ldots, i\_k \tag{3}$$

where $\mathcal{I}$ denotes the set of KB results that satisfy the query conditions, and k is the number of matching instances.

The internal logic of the function typically involves applying a filtering mechanism over a structured KB, using the parameter values to constrain the search. This may include exact matching for clearly specified slots, partial or relaxed matching to accommodate minor user input variations, or more complex domain-specific heuristics to determine relevance. The flexibility of each function depends on the domain requirements and the granularity of the KB entries it operates over.

This execution step isolates only the information necessary to fulfill the user's request, allowing the system to respond with targeted and contextually relevant content.

3.5 Result Evaluation and Prompt Augmentation

After executing the function call and retrieving a set of results $\mathcal{I} = \{i_1, i_2, \ldots, i_k\}$, the system evaluates the outcome to determine how to proceed with the response generation. This evaluation hinges on the number of retrieved instances k, which is compared against a predefined upper threshold T to assess the informativeness and usefulness of the result set.

- **No Results ($k = 0$):** If the query returns no matching instances, the system avoids generating a misleading or hallucinated response. Instead, it augments the prompt with a message that informs the model of the lack of results and guides the generation accordingly. For example, the following instruction is added to the prompt:

  ```
  No entities in the knowledge
  base match the user query.
  Suggest that the user
  reformulate or broaden their
  request.
  ```

- **Too Many Results ($k > T$):** When the number of results exceeds a predefined threshold, the system avoids overwhelming the generation with an unmanageable list. Instead, it augments the prompt with a directive indicating the overload and advising the model to guide the user toward refining their query. For example:

  ```
  The query returned too many
  entities.  Ask the user to
  provide additional constraints
  to narrow down the search.
  ```

- **Acceptable Number of Results ($0 < k \leq T$):** If the result set is within an acceptable range, the system proceeds to build a knowledge-grounded prompt. This prompt contains a textual rendering of the retrieved instances, formatted in a way that allows the LLM to generate a coherent and factually grounded response. For example, the augmented prompt might begin with a sentence like:

  ```
  Your answer should strictly
  rely on the following
  Knowledge Base:
  ```

  followed by a structured summary of the instances in $\mathcal{I}$.

This stage ensures that the system remains faithful to the available knowledge, avoids generating unsupported statements, and adapts its behavior depending on the informativeness of the KB query results. By carefully controlling when and how the KB content is introduced into the prompt, the system maintains both factual accuracy and conversational relevance.

3.6 Conversation Management and Context Trimming

To maintain continuity across turns, the system preserves the full dialogue history H =

$\{h_1, h_2, \ldots, h_n\}$, including user and system utterances as well as the initial system prompt. However, since large language models operate within a fixed context window of size C, the system must ensure that the input remains within this limit.

To do so, it dynamically monitors the length of the conversation $L(H)$, and if the combined content of the history and the current input exceeds C, the earliest dialogue turns are discarded. This process is formalized as follows:

$$\text{If } L(H) > C, \quad \text{remove } h\_1, h\_2, \ldots, \text{ until } L(H) \leq C \quad (4)$$

The trimming procedure prioritizes the retention of the most recent and relevant turns, along with the system prompt, which provides high-level behavioral instructions to the model. This ensures that the system continues to generate contextually grounded and coherent responses, even in extended interactions.

4 Experimental Setting

This section presents the experimental setup used to assess the effectiveness of our function calling approach for knowledge retrieval in Task-Oriented Dialogue systems. Specifically, we compare two strategies: (I) a baseline that retrieves the entire knowledge base for every dialogue turn regardless of context (Section 4.1) and (II) our proposed method, which selectively invokes a function to retrieve only relevant KB entries based on the user query (Section 3).

The goal of this comparison is to determine whether dynamically deciding when and what to retrieve, through structured function calls, offers a meaningful advantage over the naïve strategy of always retrieving the full KB.

In each case, the system is tasked with producing factual, contextually appropriate, and domain-specific responses. The subsections that follow detail the full-KB strategy (Section 4.1), the structure of the dataset and the underlying KB (Section 4.2), the specification of the LLM used (Section 4.3), the evaluation metrics employed (Section 4.4), and the results of the comparative analysis (Section 4.5). We also include an error analysis (Section 4.6) and a supplementary study on the computational efficiency of both approaches (Section 4.7).

4.1 Full-KB Approach

The full-KB approach embeds the entire knowledge base directly into the system's prompt. This initial prompt begins with an instruction block that defines the system's behavior, tone, and style, followed by a directive to rely exclusively on the provided KB when responding to domain-specific user queries. The complete KB remains constant throughout the interaction and ensures that the model has access to all KB entries at every step of the conversation.

As the dialogue unfolds, each user message is appended to the context, and the system generates a response based on the full prompt, comprising the instruction block, the complete KB, and the accumulated dialogue history. Each new system message is then added to the context in turn.

To stay within the model's context window, the system trims the oldest turns in the dialogue history when necessary. This trimming follows the same strategy described for the function-calling approach (see Section 3.6), ensuring that recent context and the system prompt remain intact.

4.2 Dataset and Knowledge Base Description

The experiments use the MultiWOZ 2.3 dataset (Han et al., 2021), a widely adopted benchmark for TODs. MultiWOZ contains multi-domain dialogues collected through crowdsourcing, covering seven different domains. It provides both dialogue transcripts and corresponding structured knowledge bases for each domain, making it ideal for evaluating knowledge-grounded dialogue models.

For this study, we focus exclusively on the restaurant domain. This choice is motivated by several factors: the restaurant domain is well-studied and rich of in domain-specific attributes (e.g., cuisine type, location, price range), it has a sufficiently large and detailed KB in MultiWOZ, and limiting the scope to a single domain simplifies the evaluation of knowledge retrieval strategies without introducing confounding domain variability.

To examine the effect of KB size on the performance of different retrieval approaches, we prepared three versions of the restaurant KB: the original, and two augmented versions with approximately twice and three times the original number of entries. The augmentation process involved synthetically generating new entries while preserving the data structure and realistic attribute distribu-

Table 1: KB-Alignment accuracy of function calling vs full-KB approach for different KB sizes

Method	Original KB	KB x2	KB x3
Full-KB Approach	0.652	0.615	0.549
Function calling	**0.833**	**0.816**	**0.802**

tions. Specifically:

- **Food, Area, Price, and Address:** Values were randomly sampled from the original KB's existing attribute sets.
- **Name:** New names were created by combining tokens randomly selected from original restaurant names.
- **Phone Number:** Simulated as random 10-digit numbers.

This augmentation strategy allowed us to scale the KB while maintaining consistency and realism. We limited augmentation to three times the original size because larger KBs would exceed the model's context window constraints, especially for the full-KB approach.

4.3 Model and Computational Details

Both the Function Calling and Full-KB approaches rely on the LLaMA-3.1 8B model for dialogue generation. LLaMA-3.1 (Grattafiori et al., 2024) is part of the Large Language Model Meta AI (LLaMA) family, specifically fine-tuned for generating coherent, context-aware responses in conversational settings. With approximately 8 billion parameters, it balances computational efficiency and linguistic capability, making it well-suited for complex task-oriented dialogue (TOD) scenarios.

Inference was conducted using a single NVIDIA A40 GPU with 48GB of memory. No additional fine-tuning or parameter modifications were applied; all model parameters and decoding settings remained at their default values.

4.4 Evaluation Metrics

To assess the performance of both approaches, we use the *KB-alignment* accuracy metric. This metric measures the degree to which the system's responses align with the information contained in the knowledge base. Each response from the system is evaluated on a binary scale: 1 if the response is entirely accurate with respect to the KB (i.e., all factual statements align with the knowledge stored in the KB), and 0 if the response contains any factual inaccuracy, such as providing incorrect details about restaurant availability, food types, or locations.

Only responses that involve KB references are included in the evaluation, excluding general conversational turns that do not involve knowledge retrieval (the turns where slot-values are not detected). The overall KB-alignment accuracy is computed as the average accuracy across all KB-relevant responses. Formally, for a set of N responses $r_1, r_2, \ldots, r_N$, the accuracy A is defined as:

$$A = \frac{1}{N} \sum_{i=1}^{N} \text{alignment}(r_i) \tag{5}$$

where $\text{alignment}(r_i)$ equals 1 if the response r_i is factually correct according to the KB, and 0 otherwise.

Determining whether a system message is fully consistent with the KB is a manual task performed by a human evaluator. This process can involve ambiguity, as some information may be implicitly inferable from the KB without being explicitly stated. In such cases, the evaluator must judge whether the inference is reasonable and grounded in the available KB content.

4.5 Experimental Results

Table 1 presents the KB-alignment accuracy for both the full-KB and function calling approaches, evaluated across three different KB sizes: the original MultiWOZ 2.3 KB, and versions augmented to double and triple the number of entries.

The results clearly show that the function calling approach consistently outperforms the full-KB baseline in all settings. With the original KB, function calling achieves an accuracy of 0.833, compared to 0.652 for the full-KB strategy. This performance gap widens slightly as the KB grows larger: at twice the original size, function calling maintains a high accuracy of 0.816, while the full-KB method drops to 0.615. When the KB is tripled in size, the

Table 2: Error Category Comparison for full-KB and function calling Approaches

Error Category	Full-KB x1	Full-KB x2	Full-KB x3	Func Call x1	Func Call x2	Func Call x3
RESTAURANT NOT FOUND BY NAME	31/36 (86.11%)	23/36 (63.64%)	24/36 (65.22%)	10/18 (55.56%)	8/16 (50.00%)	12/12 (100.00%)
RESTAURANT NOT FOUND BY ATTRIBUTES	20/30 (66.67%)	30/30 (100.00%)	16/19 (84.21%)	13/17 (75.41%)	27/29 (92.96%)	14/16 (87.50%)
INVALID RESTAURANT NAME	6/33 (18.18%)	11/49 (22.47%)	12/41 (29.47%)	2/29 (6.87%)	1/17 (5.83%)	1/19 (5.23%)
INVALID RESTAURANT BY ATTRIBUTES	7/26 (27.27%)	13/73 (17.65%)	2/63 (3.17%)	2/29 (6.98%)	1/79 (1.27%)	1/29 (3.49%)
INCORRECT RESTAURANT COUNT	0/0 (0%)	0/0 (0%)	0/0 (0%)	3/3 (100.00%)	0/0 (0%)	0/0 (0%)

gap becomes even more pronounced, with function calling at 0.80 and the full-KB approach falling further to 0.549.

These results suggest that the function calling mechanism, by retrieving only relevant KB entries on-demand, is more robust to increases in knowledge base size. In contrast, the full-KB method suffers from degraded accuracy as the context window becomes increasingly saturated with irrelevant information. This confirms the hypothesis that selectively retrieving information (both what and when) leads to better factual consistency than indiscriminately including the entire KB in the prompt.

Furthermore, the relatively stable accuracy of the function calling approach across KB sizes indicates that it scales more gracefully, maintaining high response quality even as the knowledge space expands. This scalability is particularly valuable in real-world scenarios, where KBs may grow over time or vary significantly across domains.

4.6 Error Analysis

In addition to evaluating KB-alignment accuracy, we performed a detailed error analysis to identify the types of errors encountered by both approaches. Errors were manually categorized based on the nature of the mismatch between the system's response and the knowledge base. The following categories emerged:

- **RESTAURANT NOT FOUND BY NAME**: The system failed to retrieve a restaurant explicitly requested by name, despite its presence in the KB.
- **RESTAURANT NOT FOUND BY ATTRIBUTES**: The system was unable to identify a restaurant matching the specified attributes (e.g., cuisine type, area, price range).
- **INVALID RESTAURANT NAME**: The response included a restaurant name that does not exist in the KB.
- **INVALID RESTAURANT BY ATTRIBUTES**: The system returned incorrect attribute values for a restaurant (e.g., stating it is in the city center when it is not).
- **INCORRECT RESTAURANT COUNT**: The reported number of restaurants matching a query was inaccurate.

Table 2 provides a detailed breakdown of the frequency of each error type for both approaches.

The full-KB approach tends to struggle more with hallucinated content, such as producing invalid restaurant names or mismatched attributes. These issues appear to worsen as the size of the KB increases, suggesting that the model's ability to attend to relevant entries diminishes when overwhelmed with a larger unfiltered context.

The function calling method, while generally more accurate, exhibits occasional retrieval failures. For example, it can miss a valid restaurant even when the name or attributes are unambiguous, indicating potential limitations in the underlying retrieval mechanism or its integration with the language model.

Overall, while function calling mitigates many of the content hallucination issues found in the full-KB approach, it introduces its own class of retrieval-related errors. Understanding these patterns is crucial for designing more robust hybrid systems that combine precise retrieval with effective language generation.

4.7 Computational Efficiency

Finally, the computational efficiency of both approaches was evaluated by comparing the input token count and the total processing time required for each dialogue. These metrics provide a practical perspective on the scalability of each method, especially as the size of the knowledge base increases.

The results, illustrated in Figure 1, show that the function calling approach is significantly more effi-

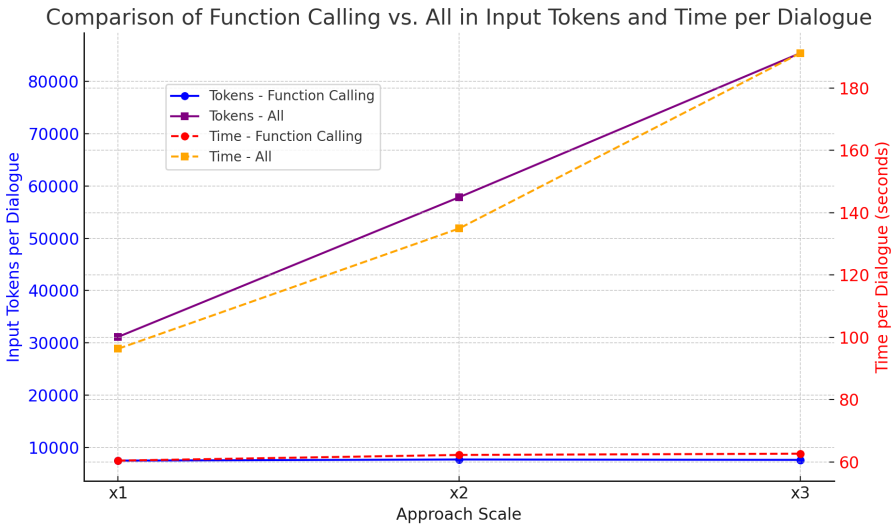

Figure 1: Comparison of Input Tokens and Processing Time per Dialogue between function calling and full-KB Approaches across KB Sizes

cient than the full-KB method across all KB sizes. By retrieving only the relevant information from the KB on demand, function calling reduces the input size passed to the language model, which in turn lowers the processing time.

For the original KB scale (x1), function calling required only 7,503 tokens and 60.5 seconds per dialogue, compared to 31,153 tokens and 96.3 seconds for the full-KB approach. This translates to a 76% reduction in prompt tokens and a 37% reduction in processing time. As the KB triples in size (x3), the difference becomes even more striking: function calling maintains a modest increase to 7,653 tokens and 64.2 seconds, while the full-KB baseline increases to over 85,000 tokens and 189.7 seconds per dialogue. This yields savings of over 90% in tokens and more than 66% in processing time. These findings confirm that function-based retrieval not only improves factual precision but also delivers substantial gains in efficiency—cutting inference time in large-KB settings, making it a highly scalable alternative for real-world end-to-end TOD deployments.

5 Conclusion

This study highlights the advantages of combining selective retrieval and function calling for improving the accuracy and scalability of task-oriented dialogue (TOD) systems. By retrieving only the necessary information from the knowledge base (KB), the function calling approach consistently delivers more accurate responses, while also reducing computational overhead and response latency. In contrast, the full-KB approach, which embeds the entire KB into each prompt, becomes increasingly inefficient and less reliable as the KB grows.

The experimental results show that function calling maintains strong KB-alignment accuracy across different KB sizes, demonstrating robustness to domain changes and scalability in growing knowledge environments. In parallel, the computational analysis reveals substantial reductions in both input token usage and processing time. This efficiency gain not only improves response latency but also reduces the overall cost of deployment, especially in resource-constrained or real-time settings.

The error analysis further reinforces these findings. As the KB size increases, the full-KB approach struggles with issues such as failing to retrieve restaurants by name, indicating that larger KBs introduce ambiguity and complexity that the model cannot easily resolve. In comparison, the function calling method demonstrates greater resilience to these challenges. The integration of selective retrieval, function calling, and prompt augmentation offers an effective strategy for building TOD systems that are accurate, efficient, and scalable. This approach provides a practical path forward for deploying dialogue agents in dynamic, knowledge-rich environments.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Si-An Chen, Lesly Miculicich, Dmitry Ustalov, Tuan Du, Adam Roberts, and Tom Kwiatkowski. 2024. Tablerag: Million-token table understanding with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ting Han, Ximing Liu, Ryuichi Takanabu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng, and Minlie Huang. 2021. Multiwoz 2.3: A multi-domain task-oriented dialogue dataset enhanced with annotation corrections and co-reference annotation. In *Natural Language Processing and Chinese Computing: 10th CCF International Conference, NLPCC 2021, Qingdao, China, October 13–17, 2021, Proceedings, Part II 10*, pages 206–218. Springer.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24(1).

Tiziano Labruna, Jon Ander Campos, and Gorka Azkune. 2024. When to retrieve: Teaching llms to utilize information retrieval effectively. *arXiv preprint arXiv:2404.19705*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Teng Lin, Yizhang Zhu, Yuyu Luo, and Nan Tang. 2025. Srag: Structured retrieval-augmented generation for multi-entity question answering over wikipedia graph. *arXiv preprint arXiv:2503.01346*.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann Lecun, and Thomas Scialom. 2023. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023. Publisher Copyright: © 2023, Transactions on Machine Learning Research. All rights reserved.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis.

Marco Roberti, Giovanni Bonetta, Rossella Cancelliere, and Patrick Gallinari. 2020. Copy mechanism and tailored training for character-based data-to-text generation. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part II*, pages 648–664. Springer.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Hassan Soliman and Iryna Gurevych. 2025. A survey on advances in retrieval-augmented generation over tabular data and table question answering. In *Proceedings of the ACL Workshop on Reasoning over Large and Graph-Structured Data (RLGMSD)*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *International Conference on Learning Representations (ICLR)*.

Xiaoxue Zang, Qian Ma, Felix Heide, Hao Zhou, Ming Shi, Paweł Budzianowski, Tsung-Hsien Wen, and Milica Gašić. 2020. Multiwoz 2.3: A multi-domain task-oriented dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI (NLP4ConvAI)*.

Yizhe Zhang, Siqi Sun, Xiang Gao, Yuwei Fang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2022. Retgen: A joint framework for retrieval and grounded text generation modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11739–11747.

Qingxia Zhao, Xiaopeng Jin, Zhibin Wang, et al. 2023. Structgpt: A structured multimodal prompting framework for task-oriented dialog. *arXiv preprint arXiv:2305.11887*.