DSRAG: A Double-Stream Retrieval-Augmented Generation Framework for Countless Intent Detection

Pei Guo*, **Enjie Liu***, **Ruichao Zhong***, **Mochi Gao***, **Yunzhi Tan**†, **Bo Hu**†, **Zang Li**
Big Data and AI Platform Department, Tencent, China
pguolst@stu.suda.edu.cn;
rzhongab@connect.ust.hk;
{karolinaliu, mochigao, boristan, harryyfhu, gavinzli}@tencent.com

Abstract

Current intent detection work experiments with minor intent categories. However, in real-world scenarios of data analysis dialogue systems, intents are composed of combinations of numerous metrics and dimensions, resulting in countless intents and posing challenges for the language model. The retrieval-augmented generation (RAG) method efficiently retrieves key intents. However, the single retrieval route sometimes fails to recall target intents and causes incorrect results. To alleviate the above challenges, we introduce the DSRAG framework combining **q**uery-to-**q**uery (Q2Q) and **q**uery-to-**m**etadata (Q2M) double-stream RAG approaches. Specifically, we build a repository of query statements for Q2Q using the query templates with the key intents. When a user's query comes, it rapidly matches repository statements. Once the relevant query is retrieved, the results can be quickly returned. In contrast, Q2M retrieves the relevant intents from the metadata and utilizes large language models to choose the answer. Experimental results show that DSRAG achieves significant improvements compared with merely using prompt engineering and a single retrieval route.

1 Introduction

Large Language Models (LLMs) (Brown et al., 2020; Hoffmann et al., 2022; OpenAI, 2022; Touvron et al., 2023) have significantly transformed the landscape of natural language processing tasks. With their robust understanding and generation capabilities, task-oriented data analysis dialogue systems have garnered widespread attention. These systems can intelligently assist data analysts (defined as users) in inquiring, analyzing, and visualizing data. One crucial aspect of these systems is intent detection (Liu et al., 2019a; Mou et al., 2022a,b; Song et al., 2023), identifying which of a fixed set of actions to take based on the user's queries. Current intent detection work typically experiments with minor intent categories (e.g., CLINC (Larson et al., 2019) datasets with 150 intents, BANKING (Casanueva et al., 2020) with 77 intents). However, in the scenario of data analysis dialogue systems, an intent consists of the metrics and dimensions that the user wishes to analyze. For example, as shown in Figure 1, **we need to detect the metric *daily active user* and the dimension *gender* from the metadata based on the user's query (more descriptions of metadata, metric, and dimension are shown in Section 3.1).** The application has numerous metrics, each with multiple dimensions, resulting in countless combinations. Therefore, traditional classification methods (Liu et al., 2019a; Bunk et al., 2020) are inapplicable. Besides, detecting the intent from countless combinations of metrics and dimensions will create a significant challenge for the model because of the limitation of long text modeling and input length.

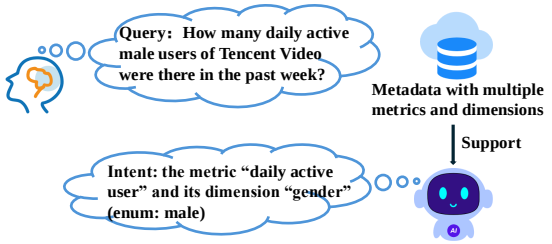

Figure 1: The intent detection example of extracting the metric and dimensions based on the user query.

To alleviate this challenge, we introduce the RAG (Lewis et al., 2020) method to retrieve key intents from the extensive pool, thereby filtering out the irrelevant intents and reducing the complexity of the problem. However, for each query, employing a single retrieval route to directly retrieve the information from the metadata that stores numerous metrics and dimensions, sometimes can't recall the target intents, causes incorrect results, and impacts

* Equal Contribution
† Corresponding Author

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Industry Track), pages 318–328

user experience. To further improve the accuracy of intent detection, we propose the comprehensive **d**ouble-**s**tream RAG (DSRAG) framework that integrates two retrieval approaches, namely **q**uery to **q**uery (Q2Q) and **q**uery to **m**etadata (Q2M). 1) Q2Q: We create a series of query templates and infill different structured metadata to build a query statement repository that simulates potential user queries. When the user's query comes, Q2Q rapidly matches it with repository statements. Once the most similar query statement is retrieved, there is no need to execute Q2M, which significantly decreases response time and error feedback probability. 2) Q2M: We employ common retrieval methods, including indexing, re-rank, etc. to obtain relevant metrics and dimensions from metadata. After that, we innovatively propose two approaches based on closed and open sourced LLMs to select the most relevant metrics and dimensions.

To validate the effectiveness of DSRAG, we conduct experiments based on real online user queries. The results show that DSRAG significantly improves accuracy from 24.7% to 78.7% compared with directly using GPT-3.5-Turbo to choose the correct intents, and from 58.7% to 78.7% and 82.7% to 90.3% compared to a single retrieval route with GPT3.5-Turbo and fine-tuned open-sourced LLMs. Our contributions are listed below:

- We show the limitations of current intent detection and a single retrieval route in data analysis dialogue systems with countless intents.
- We develop the DSRAG framework, which adopts a double-stream retrieval strategy. For a query, DSRAG first employs Q2Q to look for a similar query in the library of query statements. If it doesn't work, Q2M is employed to retrieve the relevant intents from metadata.
- The experiments on the actual online user queries show that DSRAG achieves significant improvements compared with using prompt engineering and a single retrieval route.

2 Related Work

2.1 Intent Detection & Discovery

Intent detection (Liu et al., 2019a; Bunk et al., 2020), which needs to classify the user's query into in-domain (IND) intents, plays a vital role in task-oriented dialogue (TOD) systems. Lin et al. (2023) leverage the in-context learning ability of LLMs to generate synthetic training data and preserve quality and diversity. In real-world settings, it's necessary to identify out-of-distribution (OOD) intents that are not in the pre-defined intents pool. OOD intent discovery (Lin et al., 2020; Mou et al., 2022a,b) clusters OOD intents into multi-group new intents using prior knowledge of pre-defined intents. Song et al. (2023) evaluate ChatGPT on ODD discovery tasks and provide a valuable analysis. However, in this paper, we aim to improve the accuracy of detecting the intents from countless combinations of metrics and dimensions.

2.2 LLMs for Structured Knowledge

A few works (Modarressi et al., 2023; Hao et al., 2023; He et al., 2024; Xue et al., 2024) have studied to augment LLMs with knowledge from the external structured knowledge bases (KBs), usually by designing the interfaces to obtain the relevant information from KBs and guiding LLMs to answer the results. For example, for the knowledge graph (KG) based question answering tasks, Ret-LLM (Modarressi et al., 2023) is designed to extract relational triples from user inputs and subsequently store them in a symbolic Knowledge Graph (KG) memory. This functionality is akin to the KG memory framework utilized by LangChain (Chase, 2022) and LlamaIndex (Liu, 2022). With the fact retriever injects only the relevant knowledge, KAPING (Baek et al., 2023) enhances the knowledge for the input question from KG directly in the input prompt of LLMs. KnowledGPT (Wang et al., 2023) employs the program of thought prompting as the retrieval process and can store knowledge in personalized KBs. In the context of databases (DB), PrivateGPT (Toro et al., 2023) is all about ensuring the security and privacy of LLM-based database applications. ChatDB (Hu et al., 2023), a framework that enhances LLMs with symbolic memory in databases, improves complex reasoning, and prevents error accumulation. DB-GPT (Xue et al., 2024) can provide context-aware responses and generate complex SQL queries built upon the RAG methods. The above works adopt the single retrieval route, but DSRAG uses a double-stream retrieval strategy to further improve performance.

3 Method

3.1 Problem Definition

Given the metadata, which consists of a set of defined IND metrics $M = \{m_i\}_{i=1}^{n}$, each metric con-

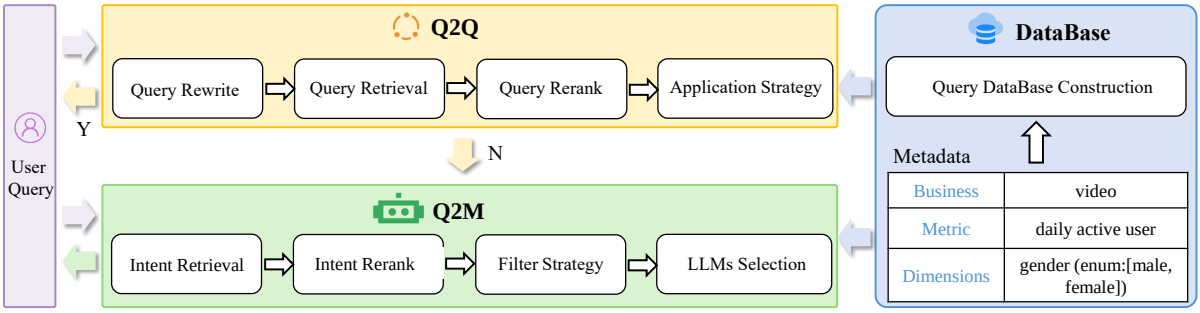

Figure 2: The overview of our DSRAG framework, comprises three parts: (i) Metadata, which consists of multiple metrics and their dimensions with enums, and we sample them to construct the query database. (ii) Q2Q, which first retrieves similar queries from the query database and returns the results with the application strategy. (iii) Q2M executes the retrieval and re-rank process based on the metadata, then utilizes LLMs to choose the most relevant metric and its dimensions with enums.

tains multiple dimensions $D = \{d^j_{m_i}\}^z_{j=1}$ and every dimension also contains zero to multiple enums $E = \{e^k_{d^j_{m_i}}\}^y_{k=0}$, we need to accurately identify the user's intent, specifically determining which metrics, dimensions and even enums are involved when a query is received. For example, as shown in Figure 2, the metric *"daily active user"* contains the dimension *"gender"*, and the enums of *"gender"* are *"male"* and *"female"*. Because the number of combinations of metrics and dimensions with different enums is multitudinous, for our experiments below, we simplify the problem by stating that each query contains one metric and zero to three potential dimensions and enums.

3.2 DSRAG Framework

As shown in Figure 2, DSRAG comprises the construction of the query database and the relevant process of Q2Q and Q2M.

3.2.1 Construction of Query Database

Based on the constructed metadata, which consists of multiple metrics and dimensions with enums, we selected 310 key metrics and 675 dimensions. Following the real online scenarios, we artificially created 132 query templates. Subsequently, we generate 1.65 million meta-samples by combining one metric with zero to two dimensions and their enums. Finally, we matched the meta-samples with the corresponding template to generate 43.25 million queries. A specific example of the construction process is shown in Appendix A.2.

3.2.2 Query to Query (Q2Q)

Based on the query database, Q2Q converts intent detection into retrieving the most similar queries.

Query Rewrite User's query statements are usually colloquial, such as *"Which TV show has been the most popular in the past week?"* To reduce the difficulty of retrieving similar queries from the query database, we need to rewrite it as *"Which TV show had the highest view counts in the past week?"* Therefore, we have defined some regular expressions to professionalize the user's query statements based on our scenario.

Query Retrieval To efficiently retrieve the relevant query statements from 43.25 million queries, we compute the relevance of the user's query and each query statement by utilizing the BM25 (Robertson and Zaragoza, 2009), which is based on weighted term frequency, and extract the top 200 highest-scoring query statements for the next stages.

Query Rerank To more accurately score the relevance between the retrieved query statements and the user's queries, we employ the cross-encoder[1] (Reimers and Gurevych, 2019), which has been proven an effective reranking approach. During training, we define a query with similar semantics to a user's query q as a positive sample q_{pos}, and vice versa as a negative sample q_{neg}. To reduce redundant information, we directly extract the metric and its dimensions from q_{pos} and q_{neg} and format them as $input_{md}$ = *"[metric] metric name [dim_name] list of dimensions (enums)"*, such as *"[metric] daily active user [dim_name] gender (male)"*. Therefore, the input format of cross-encoder is *"q [SEP]* $input_{md}$*"*, where *[SEP]* is a special division token. The classification labels

[1] https://github.com/UKPLab/sentence-transformers/tree/master

Instruction:
Based on the user's query, select the metric and dimensions that meet the query requirements from the candidate answers:
(1) If the answer only contains an metric, output the position of the metric, for example, m1
(2) If the answer contains both an metric and dimensions, output the position of the metric_dimension ID, split with "," if multiple dimensions are needed, such as m1_1001 or m1_1001,m1_1002
(3) If there is no suitable metric or dimension, output 'no answer'.

Input:
User query: How many daily active devices do male users of Tencent Video have?
Optional answers:
metric m1: average active days (dimension 18: active user type | dimension 110: gender [enum: male])
metric m2: DAU, known as: daily active user (dimension 110: gender [enum: male] | dimension 20: third-level terminal name [enum: PC])
metric m3: active device distribution (dimension 12: city | dimension 16: education)
no anwer

Output:
m2_110

Figure 3: An example of instruction tuning for open-source LLMs.

are 1 and 0 for the samples from q_{pos} and q_{neg} respectively. Besides, we select some queries from the query database and retrieve the top 100 relevant metadata for each. If the metadata matches the user's query, it's a positive sample; otherwise, it's a negative sample. Finally, we construct 2.88 million training samples based on the above approach. The triplet loss function (Schroff et al., 2015) is employed to train our cross-encoder. During inference, we follow the format $input_{md}$ to combine q and the metadata retrieved from the previous process and compute the reranking scores.

Application Strategy After the above processes, we filter out the queries whose confidence is lower than a threshold α and the final strategies are the following: 1) If all queries are filtered out, we turn to the Q2M process. 2) If only one query remains, we extract its metric and dimensions with enums to the user. 3) If multiple queries remain, we offer the top 3 options for users to choose from.

3.2.3 Query to Metadata (Q2M)

Because it is impossible to enumerate all metadata combinations, Q2M utilizes the RAG methods to retrieve multiple sets of the relevant intent and employs LLMs to choose one set.

Intent Retrieval & Rerank We first split users' queries into words with the IK Analysis plugin[2], and adopt the BM25 algorithm to calculate the relevant score between users' queries and each metric and its dimension. After that, we select the top 100 intents and rerank them using the cross-encoder introduced in 3.2.2.

Filter Strategy We design some strategies to filter irrelevant metadata. 1) Top 10 metadata are selected based on the BM25 and reranking scores, respectively. 2) Metadata with the BM25 score below β is filtered. β is set to 100 in our experiments.

LLMs Selection After the above processes, LLMs as the selectors, aim to select the most suitable metric with dimensions from the remaining candidates. We innovatively designed two methods: one approach is applying closed-source LLMs based on prompt engineering, while another is training open-source LLMs. For the first approach, we adopt the dual-step strategy, LLMs take the lead in selecting the most relevant metric, and choose the dimensions mentioned in the query (both examples of the prompts are presented in Appendix A.3 respectively). However, when the correct metric with dimensions is not in the candidates, LLMs tend to output an incorrect intent rather than answering 'There is no correct answer'. Besides, considering enterprise data privacy and security, as well as the challenge that LLMs suffer from understanding specific domain data, it's necessary to train open-source LLMs with special domain data to alleviate these challenges. Therefore, for the second method, we train an open-source LLM with LoRA tuning (Hu et al., 2022), and the training and inference sample is presented in Figure 3. Specifically, to mitigate potential hallucinations, such as outputting the unknown metric and dimension names,

[2] https://github.com/infinilabs/analysis-ik

Type	Ratio (Training - Test)
No Answer	18.2% - 13.3%
One m	41.5% - 7.3%
One m with one d	28.1% - 46.3%
One m with two or more d	12.2% - 33.0%

Table 1: The ratios of different types of samples for the training and test sets. m and d denote the metric and dimension respectively.

we require LLMs to output metric position and dimension ID. What's more, LLMs are trained to return *'No Answer'* when there is no correct intent in the candidates.

4 Experiments

4.1 Experimental Settings

Datasets To effectively train open-sourced LLMs, we collect 1592 real users' online queries about the video domain from our data analysis dialogue system. Q2M process is employed to obtain the relevant intent candidates, including intent retrieval, rerank, and filter strategy. After that, we annotate the target intents artificially based on the candidates. To improve the robustness of LLMs, we randomly shuffle the order of candidates, thereby expanding each sample to 4. Finally, there are 6368 samples in the training set. To evaluate the DSRAG framework, we also collect 300 samples from online user requests as the test set. The specific ratios of different types of samples for the training and test sets are shown in Table 1. It's noticed that *'No Answer'* indicates no metrics and dimensions are related to the query in the intent candidates. Therefore, DSRAG should respond with *'No Answer'* for these samples as unknown intents.

Evaluation Metrics We adopt two ranking metrics, namely the Hit Ratio (HR@N) and Normalized Discounted Cumulative Gain (NDCG@N) (He et al., 2017b,a) to evaluate the performance of intent retrieval and reranking. N is set to 1 to 10 for comparison. Accuracy, which means selecting the correct metrics and dimensions, is employed to assess the general performance of all processes.

Implementation For the thresholds α in Section 3.2.2 Application Strategy, we set it to 0.85 based on online scenarios. Besides, RoBERTa (Liu et al., 2019b) is employed as the reranking cross-encoder backbone. In Section 3.2.3 LLMs Selection, we utilize GPT3.5-Turbo as the selector for the first approach and open-source LLMs (LLama3-8B-IT [3] (AI@Meta, 2024) and Qwen2-7B [4] (Yang et al., 2024)) with supervised fine-tuning (SFT) for the second. The details of training hyperparameters about cross-encoder and open-source LLMs are shown in Appendix A.1.

Methods	Selectors	Accuracy (%)
Prompt Engineering	GPT3.5-Turbo	24.7
DSRAG w/o Q2M	-	38.7
	Without Training	
DSRAG	GPT3.5-Turbo	**78.7**
w/o Q2Q		58.7
	With Training	
DSRAG	Qwen2-7B-SFT	**90.0**
w/o Q2Q		82.3
DSRAG	LLama3-8B-SFT	**90.3**
w/o Q2Q		82.7

Table 2: The accuracy of different methods with two selectors on the intent detection test set.

4.2 Baselines

To evaluate the necessity to filter out the irrelevant intents, we choose the target intent with 29 non-relevant intents to form the candidates and utilize GPT3.5-Turbo to select the label intent. **The relevant prompts are presented in Appendix A.3.** Because the application has numerous metrics with multiple dimensions, resulting in countless combinations, traditional classification methods (Liu et al., 2019a; Bunk et al., 2020) are inapplicable.

4.3 Main Results

The experimental results are listed in Table 2 and can be summarized as follows: 1) It's challenging for LLMs to select the correct intents from the numerous candidate intents based on prompt engineering, which merely achieves an accuracy rate of 24.7%. Combined with the Q2M method, GPT3.5-Turbo achieves a 34% accuracy improvement (58.7% vs. 24.7%), demonstrating the effectiveness of the Q2M process in filtering out irrelevant intents. 2) Compared with prompt engineering, tuning open-source LLMs significantly improves performance, respectively achieving 82.3% and 82.7% for Qwen2-7B-SFT and LLama3-8B-

[3] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[4] https://huggingface.co/Qwen/Qwen2-7B-Instruct

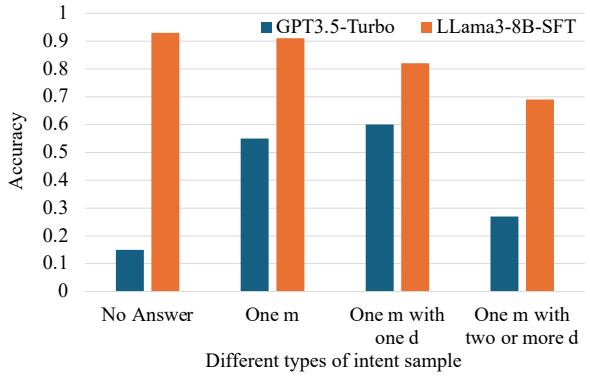

Figure 4: Comparison of GPT3.5-Turbo and LLama3-8B-SFT on the accuracy of different types of samples. m and d denote the metric and dimension respectively.

SFT. This denotes that adopting domain-specific data instruction tuning is an effective method to alleviate LLMs' insufficient understanding of domain data. 3) The complete DSRAG with different selectors achieves the best performance but significantly drops without the Q2Q process. We further statistic the coverage ratio of the Q2Q on the test set and find that Q2Q answers 39.3% samples with an accuracy of 97.4%. The above statistical results demonstrate the benefits of using DSRAG with a double-stream retrieval strategy. 4) To evaluate the efficiency of Q2M and Q2Q processes, we tested them on 40-core CPUs and an A10 GPU and found that they only require an average calculation time of 1s and 12ms respectively. Consequently, Q2Q consumed a very short time, yet brought a significant performance improvement. Overall, the DSRAG can maintain great effectiveness and efficiency.

4.4 Ablation Study

Accuracy of Different Types of Samples We further analyze the accuracy of two LLMs selection strategies employed by the Q2M module on different types of samples. As shown in Figure 4, the accuracy of prompt engineering (GPT3.5-Turbo) is only 15% on *'No answer'* samples, indicating that LLM struggles to effectively determine whether a correct answer exists and tends to output one of the intents. At the same time, the accuracy on difficult samples, which contain one metric and two or more dimensions, is only 27%. For LLama3-8B-SFT, it can effectively determine whether a correct answer exists (the performance of *No answer* sample reaches 93%) and can achieve close to 70% accuracy even on difficult samples.

Module	Metrics				
	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
Intent Rerank	0.637	0.790	0.721	0.847	0.739

Table 3: Performance of reranker in Q2M process.

Performance of Reranking In the Q2M process, the metrics with dimensions retrieval provide massive potential intents, and the ranker is employed to reorder them further. To evaluate the performance of the ranker, we adopt HR@N and NDCG@N (N is set to 1, 5, and 10) to test it. As shown in Table 3, the ranker achieves excellent performance across all metrics, which is beneficial to filter out numerous irrelevant intents, allowing LLMs to pay more attention to the top N intents.

Extensibility of Q2M To evaluate the extensibility of Q2M with the fine-tuned LLM, we conducted experiments on a news domain dataset, which comprised 100 test samples collected from our dialogue system. It's noteworthy that the metrics and dimensions in these samples never appear in the training set. We also perform intent retrieval and reranking processes and LLama3-8B-SFT is employed to select the final intent. The results show that Q2M module achieves an 87% accuracy, demonstrating its adaptability in intent retrieval and reranking, as well as the LLM's strong understanding of intent detection tasks and its ability to generalize.

5 Conclusion

In this paper, we outline the challenges of current intent detection methods. Specifically, in data analysis dialogue systems, intents are formed by combining various metrics and dimensions, resulting in countless intents and posing challenges for current works. Besides, although employing RAG approaches is effective in retrieving key intents, sometimes it can't recall the target intent. Therefore, to further improve the accuracy of intent detection, we have developed the DSRAG framework, which uses a double-stream retrieval strategy. When the query comes, Q2Q are implements to look for a similar query in the library of query statements constructed by the key metrics and dimensions with the query templates. If it doesn't find a relevant query, Q2M is employed to retrieve the relevant metrics and dimensions from metadata. The experiments on real user queries confirm that Q2Q can address a large portion of the queries with high accuracy and low latency. Additionally, the DSRAG shows sig-

nificant improvements compared to merely using prompt engineering and RAG methods.

6 Limitation

In this section, we present several of the limitations of this paper. Firstly, as shown in Table 3 of the paper, we find that HR@10 and NDCG@10 achieve 0.847 and 0.739 respectively, which means that a few correct intents are not retrieved, how to retrieve intents more accurately from metadata is one of the optimization directions. Moreover, as shown in Figure 3, we design the metric position with dimension ID or 'no answer' as the outputs, which may cause LLMs not to understand why the metric and dimension were selected, or why the output is 'no answer'. Adding explanations like the CoT approach to assist LLMs is another direction to improve performance further.

References

AI@Meta. 2024. The llama 3 herd of models. *Preprint arXiv:2407.21783*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 78–106. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and et al. 2020. Language models are few-shot learners. In *Advances in neural information processing systems*, pages 33:1877–1901.

Tanja Bunk, Daksh Varshneya, Vladimir Vlasov, and Alan Nichol. 2020. Diet: Lightweight language understanding for dialogue systems.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*.

Harrison Chase. 2022. Llamaindex.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in neural information processing systems*.

Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017a. Translation-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys*, pages 161–169.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017b. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182. International World Wide Web Conferences Steering Committee.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego De, Las Casas, Lisa Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, GeorgeVanDen Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Patrick S. H. 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 *Advances in neural information processing systems*, pages 9459–9474.

Ting-En Lin, Hua Xu, and Hanlei Zhang. 2020. Discovering new intents via constrained deep adaptive clustering with cluster refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8360–8367.

Yen-Ting Lin, Alexandros Papangelis, Seokhwan Kim, Sungjin Lee, Devamanyu Hazarika, Mahdi Namazifar, Di Jin, Yang Liu, and Dilek Hakkani-Tur. 2023. Selective in-context data augmentation for intent detection using pointwise V-information. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1463–1476. Association for Computational Linguistics.

Jerry Liu. 2022. Llamaindex.

Jiao Liu, Yanling Li, and Min Lin. 2019a. Review of intent detection methods in the human-machine dialogue system. In *In Journal of physics: conference series*, volume 1267.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *Preprint arXiv:1907.11692*.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2023. Ret-llm: Towards a general read-write memory for large language models.

Yutao Mou, Keqing He, Yanan Wu, Pei Wang, Jingang Wang, Wei Wu, Yi Huang, Junlan Feng, and Weiran Xu. 2022a. Generalized intent discovery: Learning from open world dialogue system. In *Proceedings of the 29th International Conference on Computational Linguistic*, pages 707–720.

Yutao Mou, Keqing He, Yanan Wu, Zhiyuan Zeng, Hong Xu, Huixing Jiang, Wei Wu, and Weiran Xu. 2022b. Disentangled knowledge transfer for ood intent discovery with unified contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2:Short Papers)*, pages 46–53. Association for Computational Linguistics.

OpenAI. 2022. Gpt-3.5-turbo.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. In *Foundations and Trends in Information Retrieval*, volume 3, pages 333–389.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiaoshuai Song, Keqing He, Pei Wang, Guanting Dong, Yutao Mou, Jingang Wang, Yunsen Xian, Xunliang Cai, and Weiran Xu. 2023. Large language models meet open-world intent discovery and recognition: An evaluation of chatgpt. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10291–10304. Association for Computational Linguistics.

Iván Martínez Toro, Daniel Gallego Vico, and Pablo Orgaz. 2023. Privategpt.

Touvron, Hugo, Lavril, Thibaut, Izacard, Gautier, Martinet, Xavier, Lachaux, Marie-Anne, Lacroix, Timoth'ee, Rozi'ere, Baptiste, Goyal, Naman, Hambro, Eric, Azhar, Faisal, Rodriguez, Aurelien, Joulin, Armand, Grave, Edouard, Lample, and Guillaume. 2023. Llama: Open and efficient foundation language models.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases.

Siqiao Xue, Caigao Jiang, Wenhui Shi, Fangyin Cheng, Keting Chen, Hongjun Yang, Zhiping Zhang, Jianshan He, Hongyang Zhang, Ganglin Wei, Wang Zhao, Fan Zhou, Danrui Qi, Hong Yi, Shaodong Liu, and Faqiang Chen. 2024. Db-gpt: Empowering database interactions with private large language models.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, and et al. 2024. Qwen2 technical report. *Preprint arXiv:2407.10671*.

A Appendix

A.1 Experiment Details

We list the main training hyper-parameters about the ranker and selector which are shown in Table 4.

Model (Role)	Cross-Encoder (ranker)	open-source LLMs (selector)
learning rate	2e-5	1e-5
batch size	64	8
LoRA dim	-	16
scheduler	WarmupLinear	Cosine
optimizer	AdamW	AdamW
warmup	288k	100
epochs	1	6
GPUs (A100)	1	2

Table 4: The details of experimental settings.

A.2 Query Database Construction

The construction process of the query database is shown in Figure 5, which combines the intents with query templates.

A.3 Prompts for intent detection

The specific prompts for GPT-3.5-Turbo to intent detection are shown in Figure 6 and 7.

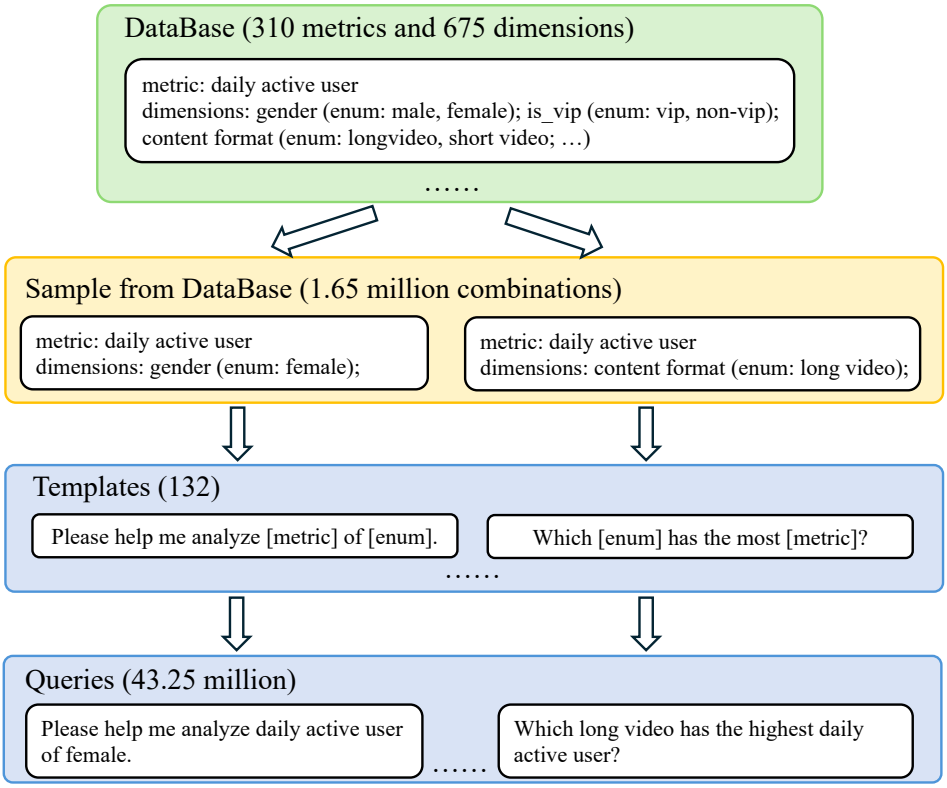

Figure 5: The process of query database construction.

Prompt:
You are a natural language processing expert and data analysis expert, you need to complete a task: receive user queries, understand the metrics and dimensions that users want, and then choose from multiple candidate answers to find one that meets the requirements correct answer.
The correct answer judging standard: the metric is consistent, and for the dimensions that the user wants, there is zero to three dimensions that can satisfy the optional dimension.
Please follow the rules**:** Just output one json, and then stop the output immediately.

Following is an example of user needs:
User query: How many active devices do male users of Tencent Video have last Wednesday?
Optional answers (in no particular order):
Candidate 1: metric: average active days; dimensions: ["active user type", "gender"]
Candidate 2: metric: DAU, known as: daily active user; dimensions: ["gender", "third-level terminal name (enum: PC)"]
Candidate 3: metric: active device distribution; dimensions: ["city", "education"]
Output: {"The metric that user wants": ["number of active devices"], "The dimension that user wants": ["male users"], "Final choice of answer group": 2}

Real user's query:
User query: {query}
Optional answers (in no particular order): {List of metadata}
Output:

Figure 6: The prompt for GPT3.5-Turbo to choose the correct metric following the user's query.

Prompt:
You are a data analysis assistant. You need to carefully and accurately analyze user queries, first extract the dimensions that users want to view, then judge whether there are dimensions that can meet the needs in the selectable dimensions, and then return the corresponding dimensions.

Please be sure to follow the guidelines below:
1. Only output one json, then stop output immediately.
2. The dimension is the user's limitation on the value of the metric: if the user's needs limit certain value ranges for the metric, then this value range is the dimension, but do not extract time and business name.
3. For each dimension of the user's question, only answer one most matching dimension.
4. Only answer the dimension name, no need to answer the explanation of the dimension

Following are some examples of user queries:
Available dimensions: ["gender: male, female, unknown", "third-level terminal name: PC”]
User query: How many active devices do male users of Tencent Video have last Wednesday?
Metric: DAU
Output: {"The dimension that user wants": ["male"], "Is there an optional dimension to meet": true, "Selected dimensions": ["gender"]}

Real user’s query:
Available dimensions: {list of dimension with enums}
User query: {query}
Metric: {metric}
Output:

Figure 7: The prompt for GPT3.5-Turbo to choose the correct dimensions with enums following the user’s query and chosen metric.