PlanRAG: A Plan-then-Retrieval Augmented Generation for Generative Large Language Models as Decision Makers

Myeonghwa Lee[*], **Seonho An**[*], **Min-Soo Kim**[†]
School of Computing, KAIST
{myeon9h, asho1, minsoo.k}@kaist.ac.kr

Abstract

In this paper, we conduct a study to utilize LLMs as a solution for decision making that requires complex data analysis. We define **Decision QA** as the task of answering the best decision, d_{best}, for a decision-making question Q, business rules R and a database D. Since there is no benchmark that can examine Decision QA, we propose Decision QA benchmark, **DQA**. It has two scenarios, Locating and Building, constructed from two video games (Europa Universalis IV and Victoria 3) that have almost the same goal as Decision QA. To address Decision QA effectively, we also propose a new RAG technique called the *iterative plan-then-retrieval augmented generation* (**PlanRAG**). Our PlanRAG-based LM generates the plan for decision making as the first step, and the retriever generates the queries for data analysis as the second step. The proposed method outperforms the state-of-the-art iterative RAG method by 15.8% in the Locating scenario and by 7.4% in the Building scenario, respectively. We release our code and benchmark at https://github.com/myeon9h/PlanRAG.

1 Introduction

In many business situations, decision making plays a crucial role for the success of organizations (Kasie et al., 2017; Gupta et al., 2002). Here, decision making involves analyzing data, ultimately leading to the selection of the most suitable alternative to achieve a specific goal (Provost and Fawcett, 2013; Diván, 2017). For example, we assume that one of the goals of the pharmacy company "Pfizer" is to minimize the production cost while maintaining on-time delivery from plants to customers in the pharmaceutical distribution network (Gupta et al., 2002), and the production cost is proportional to the amount of operation time and number of employees of a plant. Then, Pfizer may face the following decision-making problems: (P1) which plant it should operate or stop, and (P2) how many employees it should hire for each plant.

In general, the decision-making task requires performing the following three steps: (1) making a plan for which kind of analysis is needed for decision; (2) retrieving necessary data using queries; (3) making a decision (i.e., answering) based on the data (Troisi et al., 2020; Sala et al., 2022). To make the Steps (2) and (3) easier, a lot of decision support systems have been developed and utilized during the past few decades (Gupta et al., 2002; Eom and Kim, 2006; Power, 2007; Hedgebeth, 2007; Power, 2008; Kasie et al., 2017). However, humans still have been in charge of the most hard part, Step (1). The goal of this study is to investigate the possibility of replacing the human role with a Large Language Model(LLM) such that it performs not only Steps (2) and (3) but also Step (1), that is, all the Steps end-to-end.

To achieve the goal, we propose, **Decision QA**, a new decision-making task for language models. Decision QA is defined as a QA-style task that takes a pair of database D, business rules R and a decision-making question Q as input and generates the best decision as output. Figure 1 shows a situation in Europa Universalis IV game where countries compete in trade at the Age of Discovery, as an example of Decision QA. Each country decides which trading city (i.e., node) it should locate a merchant on to maximize its profit on its main (i.e., home) trading node. The example shows that a decision-making LLM decides to locate a merchant in Doab to maximize the profit of Deccan, the home trading node of the country BAH, after analyzing the database about international trade.

Next, we propose a benchmark for Decision QA called **DQA**, which consists of the following two scenarios: Locating and Building. The former scenario consists of decision questions like "Which

---

[*]Equal contribution.
[†]Corresponding author.

Where should I locate my merchant 🧑 ? My goal is maximizing **BAH**'s profit on home node, **Deccan**.

<The database about the international trade>

Country table

country	home node	development
BAH	Deccan	186.5
JNP	Doab	143.7
BNG	Ganges	143.7
⋮	⋮	⋮

NodeCountry table

trade node	country	is home	$TP_{country}$	has merchant
Deccan	BAH	True	160	False
Doab	BAH	False	71	False
Ganges	BAH	False	0	False
⋮	⋮	⋮	⋮	⋮

TradingNode table

trade node	is inland	LV	IV	OV	TP_{total}
Deccan	True	8.91	0.82	2.171	1128
Doab	True	6.98	0.87	1.564	1243
Ganges	False	8.31	1.07	1.647	1172
⋮	⋮	⋮	⋮	⋮	⋮

TradingFlow table

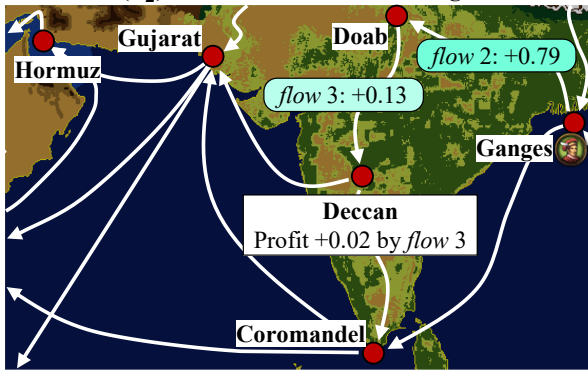| source	destination	flow
Doab	Deccan	0.78
Ganges	Doab	0.82
⋮	⋮	⋮

Step 1: Determine available decisions by finding source nodes. Step 2: Ascertain flow increments by decisions. Step 3: Calculate profit increments by decisions.

Step 2: Retrieving data and analyze it

Decision 1 (d_1): Locate a merchant to Doab.
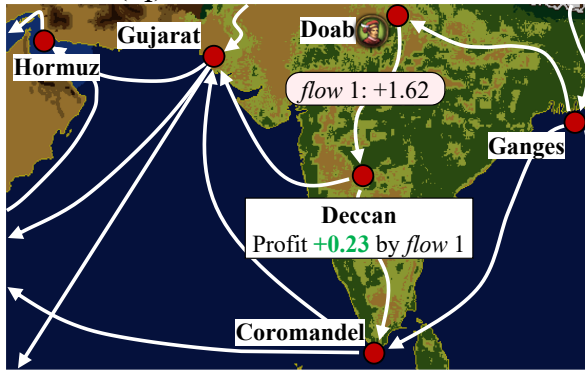
Decision 2 (d_2): Locate a merchant to Ganges.

Step 3: Answering based on the result of data analysis

You should locate your merchant to the **Doab** node to steer value, so that maximize profit of BAH.

Figure 1: Example of Decision QA. A red dot represents a trading node. A *Profit* in the Deccan box indicates a potential profit change by each decision. Note that the potential profit changes are not in the database, which should be calculated from the database. Each country has only a single main(home) trading node. The underlined column names in a table indicate the key of the table.

trade node should I locate a merchant on" (similar to P1 of Pfizer). The latter consists of questions like "How many woods should I supply to a factory" (similar to P2 of Pfizer). Due to the difficulty of building DQA using real-world business data, we built the benchmark by extracting the game data involving 301 specific situations from the two video games Europa Universalis IV and Victoria 3[1], which well imitate real business situations. To eliminate the randomness of the game and publish our benchmark, we develop the game simulators that record the decision results for 301 situations. We

utilize the results as annotations for the questions of DQA.

The recent breakthrough in LLMs is making it possible to replace Steps (2) and (3) of the decision making task with LLMs, in particular, based on *Retrieval-Augmented Generation* (RAG) technique. So far, a lot of RAG-based methods have been proposed for various tasks(Lewis et al., 2020; Khandelwal et al., 2020; Izacard and Grave, 2021; Borgeaud et al., 2022; Izacard et al., 2023; Yasunaga et al., 2023; Jiang et al., 2022a; Shi et al., 2023). In these methods, a retriever finds external data highly relevant to a question and conveys it to

[1]Grand strategy games published by Paradox Interactive

LMs, so that LMs can generate an answer based on it (Lewis et al., 2020). Recently, the *iterative RAG* technique has also been proposed to address more complex problems that should utilize retrieved results to perform further retrievals (Trivedi et al., 2023; Jiang et al., 2023b).

However, the existing RAG-based methods mainly focus on the knowledge-based QA tasks (Karpukhin et al., 2020; Trivedi et al., 2023), but do not focus on the decision making QA task. As a result, they are not very good at handling Step (1), i.e., making a plan for decision, in our observation. For instance, in Figure 1, an LM for decision making should reason which analysis is needed to perform for maximize the profit of Deccan. However, the existing methods just try to identify, for example, what Deccan is.

To address this limitation, we propose the iterative plan-then-retrieval augmented generation technique, **PlanRAG**, which extends the iterative RAG technique for Decision QA. A PlanRAG-based LM first makes a plan for which kind of analysis it needs by examining data schema and questions (the *planning* step). Next, it retrieves the scattered pieces of data for the analysis by generating and posing queries (the *retrieving* step). Finally, it assesses whether it needs to make a new plan for further analysis, and then repeats both the planning and retrieval steps iteratively (the *re-planning* step), or makes a decision based on the data (the *answering* step).

To validate the effectiveness of our PlanRAG on Decision QA, we compare both the state-of-the-art iterative RAG-based LM and our PlanRAG-based LM for the DQA benchmark. Our contributions are summarized as follows:

- We define a new challenging task, **Decision QA**, which requires both planning and data analysis for decision making.

- We propose the benchmark for Decision QA called **DQA** having two scenarios of Locating and Building.

- We propose a new retrieval-augmented generation technique, **PlanRAG**, that can significantly enhance the capability of decision making of LLMs.

- We demonstrated that PlanRAG is far more effective than the iterative RAG technique for Decision QA.

2 Decision QA task

We define **Decision QA** as the task of answering the best decision d_{best} given a decision-making question Q, business rules R, and structured database D that follows a schema S. Here, Q contains a textual goal that users want to be achieved through the decision d_{best}, and R contains textual descriptions of formulas that are referenced to reason d_{best}.

Without loss of generality, the database D is too large to fit in the input of an LM all at once. Therefore, we assume that an LM retrieves data from D by posing a query for data analysis, which we call as a *data analysis query* hereafter.

We assume that D is either a *labeled property graph (LPG) database* or a *relational database (RDB)*. For example, the database in Figure 1 is a relational database. Here, an LPG refers to a graph with properties on edges and nodes, which is widely used in the industry (Akoglu et al., 2015; Guo et al., 2020). We call a labeled property graph database as simply a *graph database*, or *GDB* hereafter.

3 DQA: Decision QA benchmark

3.1 Locating scenario

The database in this scenario is composed of the following four tables when D is RDB. It also can be easily represented as GDB by regarding the tuples in TradingFlow and NodeCountry as edges and the tuples in TradingNode and Country as vertices.

Table name	Column name	Meaning
Country	country	country name (e.g. BAH)
	home node	home node of a country
	development	economical size of the country
TradingNode	trade node	trade node name (e.g. Doab)
	LV(local value)	amount of value produced inside the node
	IV (incoming value)	the amount of value coming from neighbor source nodes (proportional to the flow between source and destination)
	TP_{total}	sum of the trading powers of all countries on this trade node
NodeCountry	trade node	trade node name
	country	country name
	$TP_{country}$	trading power of the country on this trade node
	has merchant	True if a merchant exists
TradingFlow	source	source trade node name
	destination	destination trade node name
	flow	flow weight from source to destination

Table 1: RDB schema for Locating. The **bold** indicate the columns which values depend on other values and a user decision. Some columns are omitted.

There are some business rules for decision making. The column values in bold are calculated from some other values in the database and a user decision (e.g., location of a merchant) according to the rules. If a user locates a merchant on a trading node for a country, the flow from the trading node to the home node of the country increases. In the below rules, c is a country, n a trading node, src a source node, $dest$ a destination node, and h the home node of c. We also denote the set of countries as C, and the set of TradingFlow tuples as F. TPR means Trading Power Ratio.

$$TPR(n, c) = TP_{country}(n, c) / TP_{total}(n) \quad (1)$$
$$IV(dest) = 1.05 \cdot \Sigma_{(src,dest) \in F} flow(src, dest) \quad (2)$$
$$profit(c) = (LV(h) + IV(h)) \cdot TPR(h, c) \quad (3)$$

In this scenario, the goal of decision making is to choose trading node that can maximize $\Delta profit(c)$ of the given target country c.

3.2 Building scenario

The database D in this scenario is composed of the following four tables when D is RDB. It also can be easily represented as GDB by regarding the tuples in Demand and Supply as edges and the tuples in Goods and Buildings as vertices.

Table name	Column name	Meaning
Goods	goods ID	ID of a goods
	BP	Base Price of goods
	CP	Current Price of goods
	PD (pop demand)	minimum amount of demand for the goods
Building	building ID	ID of a building
	level	size of the building
Supply	goods ID	goods ID
	building ID	building ID
	MS	Maximum amount of the goods that could be produced by the building
	CO	Current amount of the goods produced by the building
	level	size of the building
Demand	goods ID	goods ID
	building ID	building ID
	MD	Maximum amount of the goods that could be consumed by the building
	CI	Current amount of the goods consumed by the building
	level	size of the building

Table 2: RDB schema for Building. Some columns are omitted.

The most basic business rule is that $CO(g, b)$ increases for any goods g, if a decision maker expands the factory building b. In the below rules, g

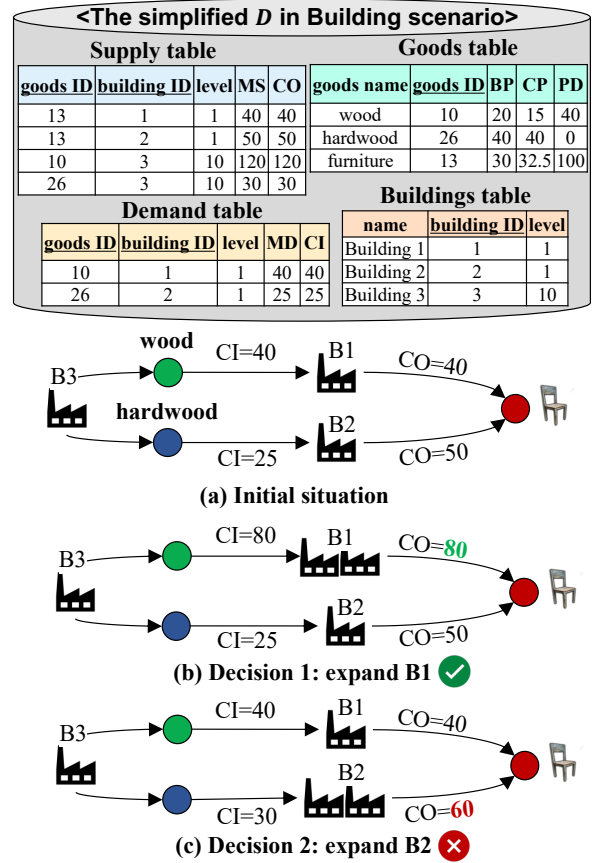| goods ID	building ID	level	MS	CO
13	1	1	40	40
13	2	1	50	50
10	3	10	120	120
26	3	10	30	30

Figure 2: Example of the Building scenario. The red, green, and blue circles indicate furniture, wood, and hardwood, respectively. We assume that the goal is reducing the price of furniture by deciding the factory between B1 and B2 to expand and so increase its production.

is goods, and b is a building. We also denote the set of Supply as Sup, and the set of Demand as Dem. TD means Total Demand, and TS Total Supply. Other major business rules are as follow:

$$TD(g) = PD(g) + \Sigma_{(g,b) \in Dem} MD(g, b) \quad (4)$$
$$TS(g) = \Sigma_{(g,b) \in Sup} CO(g, b) \quad (5)$$
$$CP(g) = BP(g) \cdot$$
$$(1 + 0.75 \cdot \frac{TD(g) - TS(g)}{max(TD(g), TS(g))}) \quad (6)$$

In this scenario, the goal is to minimize $CP(g)$ for a given goods g.

3.3 Statistics of DQA

DQA consists of a total of 301 pairs of D and Q: (1) 200 pairs for the Locating scenario and (2) 101 pairs for the Building scenario. Each database again has two versions, RDB and GDB, for the

same corresponding question, and so, a total of 602 databases are provided in the DQA benchmark. We assume that SQL is used for RDB, while Cypher Query Language (CQL) (Francis et al., 2018) used for GDB. Table 3 shows some statistics of the databases in DQA. Details for data collection are in Appendix A.1.

Statistics	Locating	Building
# of $\langle Q,D \rangle$ pairs	200	101
Relational DB (RDB)		
Avg. rows in tables of D	2038.8	579.0
Avg. cols in tables of D	4.5	4.5
Graph DB (GDB)		
Avg. # of edges of D	1432.3	374.7
Avg. # of vertices of D	606.5	204.3

Table 3: Statistics about the databases in DQA.

4 Methodology: PlanRAG

For Decision QA, the existing RAG technique (Jiang et al., 2023b; Trivedi et al., 2023) tries to answer the best decision d_{best} for given $\langle Q,S,R \rangle$ through a single type of reasoning that utilizes on the results retrieved from D by data analysis queries. Figure 3 (a) shows its reasoning process. If the retrieval from D is performed only once, the process is called as *single-turn RAG*. Otherwise, if the retrieval is performed multiple times, the process is called as *iterative RAG*.

In contrast, our *iterative plan-then-retrieval augmented generation* (PlanRAG) technique tries to answer d_{best} through two types of reasoning. The first type of reasoning is making a plan, and the second type of reasoning is similar to the reasoning of the existing RAG, i.e., answering based on the results retrieved from D by of data analysis queries. In particular, we built a single LM that can perform both types of reasoning because to reduce the side effects of using separate LMs. We prompted the LM by adding the 'Plan' and 'Re-plan' instructions to ReAct (Yao et al., 2023), about which details are in Appendix. A.6.2. Figure 3 (b) shows the reasoning process of PlanRAG, where we explain the steps of (1) planning, (2) retrieving & answering, and (3) re-planning, in detail as follows.

Planning: In this step, an LM takes $\langle Q, S, R \rangle$ as input, and then generates an initial plan for data analysis. The initial plan describes a series of data analyses that needs for decision making and so need to be performed in the retrieving step. The red box in Figure 4(b) shows its example.

Retrieving & Answering: Unlike the previous RAG technique, an LM takes not only $\langle Q, S, R \rangle$ but also the initial plan as input. Then, it can generate data analysis queries for decision making much more effectively than the previous RAG. Figure 4 shows how PlanRAG-based LM generates the queries differently from the previous RAG. The queries are actually executed by SQL or Cypher to the database through RAG interfaces such as LangChain[2] and LlamaIndex[3]. The query results are used iteratively for reasoning about whether it needs re-planning or just a further retrieval for better decision making. Through the backward link to the planning process, the planning and retrieving processes are iteratively performed until an LM determines that there is no longer a need for further analysis to make a decision.

Re-planning: Re-planning is done when the initial plan is not good enough to solve the decision question. In order to make the LM possible to decide whether re-planning or not, we prompted the LM with some instructions to assess the current plan by referring to the result of each retrieval step (see Appendix. A.6.2 for details). As a result, the LM takes not only $\langle Q, S, R \rangle$ but also a current plan and query results as input and generates a new plan to do further analysis, or correct the direction of previous analysis.

5 Experiments

5.1 Experimental Setup

To validate the effectiveness of the proposed PlanRAG for the Decision QA task, we implement and compare four different decision-making LMs: (1) SingleRAG-LM based on Single-turn RAG, (2) IterRAG-LM based on Iterative RAG, (3) PlanRAG-LM based on PlanRAG, (4) PlanRAG-LM w/o RP, which means PlanRAG without re-planning. Prompts based on ReAct (Yao et al., 2023) are used to make these LMs as decision makers (see details in Appendix A.6). All these decision makers are implemented by GPT-4 (OpenAI, 2023) with a zero temperature and the LangChain library. In terms of database, we use MySQL[4] for an RDBMS and Neo4j[5] for a GDBMS.

All experiments are conducted in a zero-shot and single run setting. This zero-shot setting is predi-
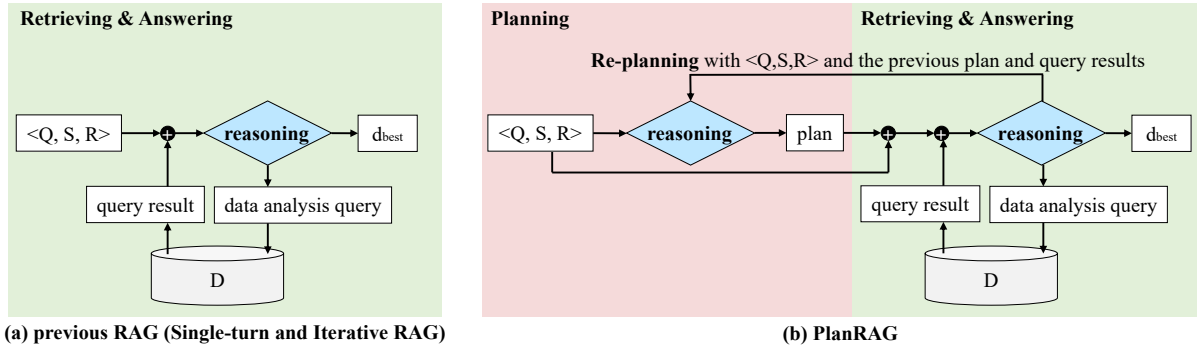

Figure 3: Comparison of the reasoning processes of between previous RAG and our PlanRAG techniques.
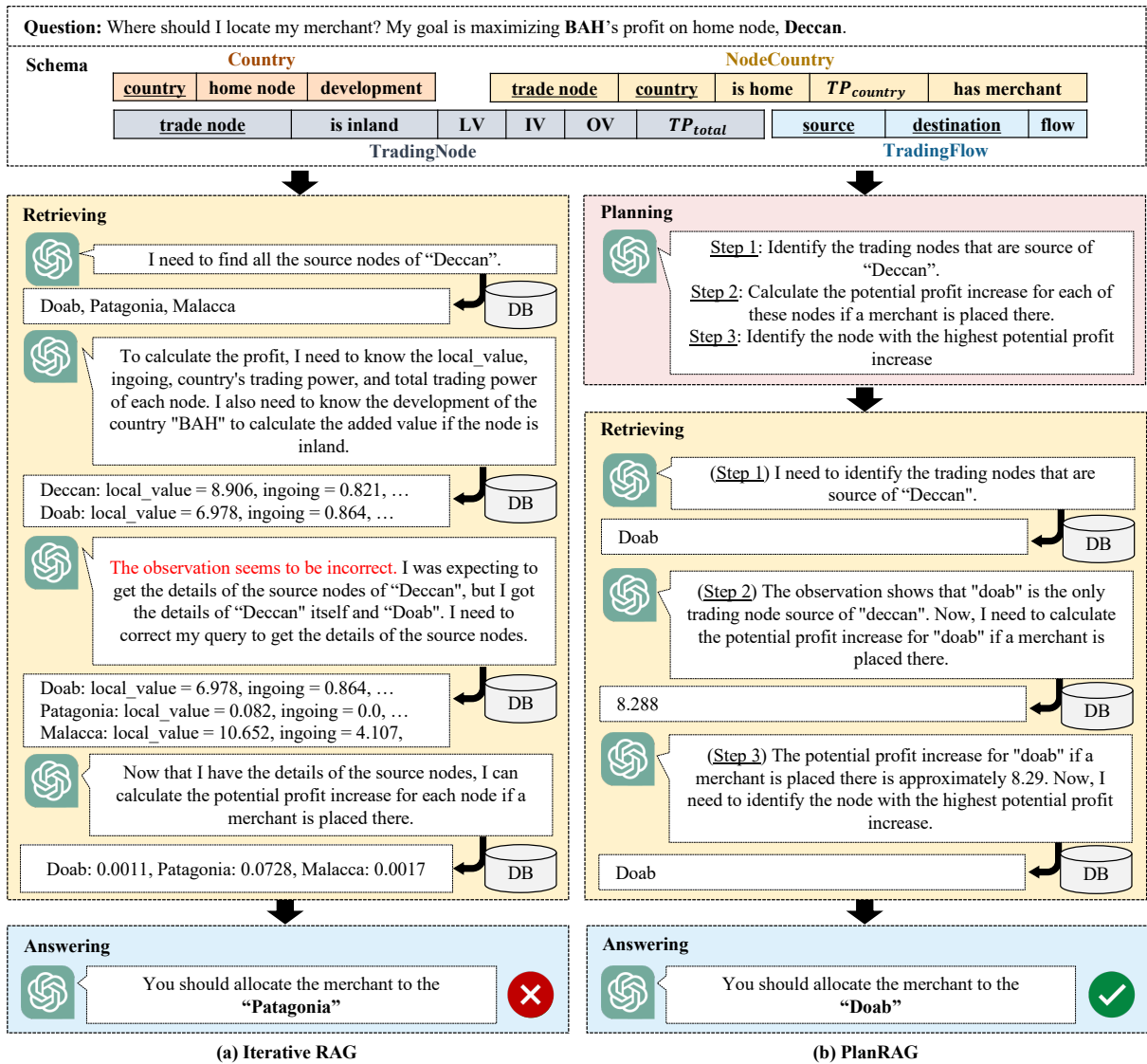

Figure 4: Example of the reasoning processes of previous Iterative RAG and our PlanRAG in the Locating scenario.

cated on the following two reasons. First, in most of real-world business situations, it is very hard to know the strategies for making the best decisions in advance. Second, in few-shot settings, we have observed that an LM is overfitted to not only the

problem-solving strategy but also the content of a database in the given shots.

An answer from a decision maker is considered correct only if it is semantically identical to a ground-truth best decision on DQA. For example,

in Figure 4 (b), we consider the answer is correct since the ground-truth best decision is *Doab*.

5.2 Results and Analysis

Main results: Table 4 presents the experimental results, where significantly improves the performance of decision making for both scenarios, by 15.8% for Locating and 7.4% for Building, compared to the existing SOTA technique, Iterative RAG (Yao et al., 2023). It shows well the effectiveness of PlanRAG for the decision making task. The reason why PlanRAG is relatively more effective in Locating than in Building is that Building requires a longer traversal than Locating scenario and it makes planning harder than Locating. The accuracy of SingleRAG-LM in Building is very low, which is because the Building scenario requires generating a very complex query that is hard to be reasoned at once. SingleRAG-LM failed to retrieve any results from the database in over 60% of Locating and 95% of Building questions. Table 4 also present that no re-planning in PlanRAG leads to a decrease in accuracy, in particular, by 10.8% in Locating and 0.9% in Building. This result shows that the re-planning process is helpful and important to the decision maker LM of the PlanRAG technique for the decision making task.

Decision makers	Locating	Building
Single-turn RAG		
SingleRAG-LM	30.5	2.5
Iterative RAG		
IterRAG-LM	48.5	37.6
PlanRAG (ours)		
PlanRAG-LM	**64.3**	**45.0**
PlanRAG-LM w/o RP	53.5	44.1

Table 4: Accuracy(%) of the techniques for DQA (Each accuracy is an average of the accuracies in RDB and GDB).

Analysis for SR and MR: There are relatively simple questions and relatively hard questions in DQA. To check the effectiveness of PlanRAG according to the degree of difficulty of questions, we divide DQA questions into two different types: (1) Single retrieval (*SR*) questions, and (2) Multiple Retrieval (*MR*) questions. Here, SR refers to the case where IterRAG-LM performs a single retrieval from D for solving the question, while MR refers to the questions that it performs multiple retrievals for solving the question. There are a total of 84 $\langle Q, D \rangle$ pairs of SR and 518 $\langle Q, D \rangle$ pairs of MR. We compare

the accuracy of IterRAG-LM and PlanRAG-LM for SR and MR, which results are presented in Figure 5.
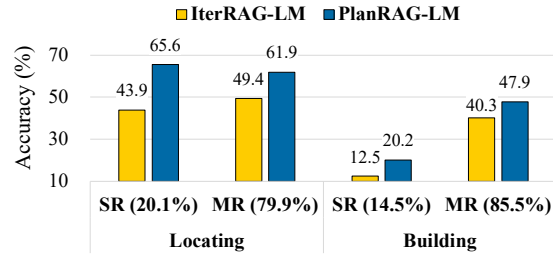

Figure 5: Accuracy(%) of IterRAG-LM and PlanRAG-LM for SR and MR questions.

In the result, PlanRAG-LM outperforms much more IterRAG-LM for the SR questions than for the MR questions. It is because the SR questions actually are not so easy in many cases. They are the ones that IterRAG-LM tried to solve only by using a single retrieval. That is, some of them are the questions that IterRAG-LM underestimated its degree of difficulty, but actually are relatively hard ones that require multiple retrievals. In contrast, PlanRAG-LM reduces the likelihood that it understands the degree of difficulty of given questions through the planning step and performs multiple retrievals according to the plan. As a result, it could significantly improve the accuracy. For the MR cases, PlanRAG-LM is still more effective than IterRAG-LM because the former performs data retrievals relatively systematically according to the plan, whereas the latter performs retrievals relatively in a disorganized manner like in Figure 4 (a).

Analysis for RDB and GDB: Table 5 presents the accuracy of LMs for two different databases, RDB and GDB, in DQA. In the results, PlanRAG-LM is more effective than other LMs in both scenarios regardless of the database types, i.e., RDB and GDB. We note that PlanRAG-LM is more effective for GDB than for RDB in the Building scenario. It is because Building is a harder scenario than Locating that requires a longer traversal in GDB (or a more number of joins in RDB) for answering a question. For example, the questions in Locating need just a single-hop traversal from source nodes to the home node, but the ones in Building need a multi-hop traversal to find high supply goods in Figure 2.

Rate of missed data analysis: In each scenario, there are some critical values that need to be queries

	Locating		Building	
Decision makers	RDB	GDB	RDB	GDB
SingleRAG-LM	25.5	35.5	2.0	3.0
IterRAG-LM	37.5	59.5	34.7	40.6
PlanRAG-LM	**64.5**	**64.0**	**40.6**	**49.5**

Table 5: Accuracy(%) of LMs for RDB and GDB.

or calculated in order to answer the questions. For example, Locating has such values including IV and TP_{total}, and Building has such values including CO and PD. In order to analyze why PlanRAG is more effective than IterRAG-LM, we measure the rate of missed data analysis for querying or calculating such values. We use IV and TP_{total} for Locating, and CO and PD for Building, as a criteria. Table 6 shows that PlanRAG-LM has low rates, 1.3% and 21.8%, while IterRAG-LM has higher rates, 3.3% and 33.2%. It means that IterRAG-LM would achieve lower accuracy than PlanRAG-LM even though it could do reasoning perfectly. According to Table 6, PlanRAG-LM might be able to achieve the accuracy of 98.7% in Locating, but the actual accuracy in Table 4 is lower than that. It is because reasoning (including planning) itself is very challenging besides missed data analysis.

Decision makers	**Locating**	**Building**
IterRAG-LM	3.3%	33.2%
PlanRAG-LM	**1.3%**	**21.8%**

Table 6: Rate of missed data analysis.

Analysis for failure cases: We classify each failure case into five error categories as follows: (1) *CAN*, which means an LM fails to solve a question by considering improper candidates (e.g. *dest* of Deccan in Figure 1) and answer them; (2) *MIS*, missed data analysis; (3) *DEEP*, using retrieved data or equations unproperly; (4) *QUR*, query generation error; and (5) *OTH*, other errors (e.g. exceeding token length limits). For example, we classify 4(a) as DEEP, because an LM misused some equations and so underestimated the profit of Doab. We compare failure cases done by IterRAG-LM and PlanRAG-LM based on these categories, in Figure 6.

In the result, PlanRAG-LM significantly reduces CAN and MIS errors for both scenarios. It means that PlanRAG-LM can understand a question of Decision QA better and query critical data for the question better than IterRAG-LM. We also note that PlanRAG-LM has slightly more DEEP cases
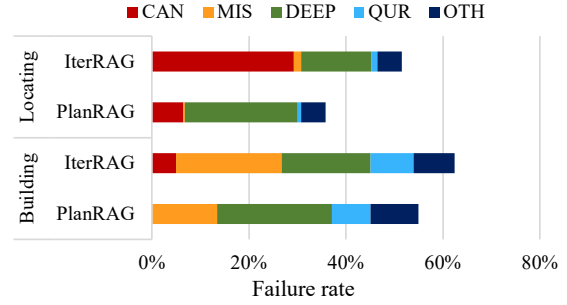

Figure 6: A result of failure case analysis. Failure rate means the number of failure questions divided by the number of all questions.

than IterRAG-LM for both scenarios. In our observation, DEEP error appears only if there are no CAN or MIS errors. For example, in Figure 4(a), there is no chance for an LM to underestimate the profit of Doab (i.e. making DEEP error) when it fails to take Doab as one of the candidates (i.e. making CAN error). Thus, we can say that the increase of DEEP cases in PlanRAG-LM comes from side effects of reducing both CAN and MIS cases.

Analysis for re-planning: PlanRAG-LM performs re-planning for some DQA questions. Table 7 presents the distribution of questions based on the number of re-plannings conducted by PlanRAG-LM and the accuracy improved by re-planning. Detailed cases and statistics for re-planning are in Appendix A.8.

No. of re-plannings	**Locating**	**Building**
0	376 (66.8%)	125 (57.6%)
1	0 (-)	0 (-)
2	12 (41.7%)	24 (41.7%)
3	5 (20.0%)	23 (21.7%)
more than 4	7 (0.0%)	30 (13.3%)
Total	400 (64.3%)	202 (45.0%)

Table 7: Statistics of DQA questions according to the number of re-plannings conducted by PlanRAG-LM. The percentage in parentheses indicates the accuracy(%) on a set of corresponding questions.

In the result, PlanRAG-LM re-plans more frequently in the Building scenario than the Locating scenario. PlanRAG-LM re-plans 24 out of 400 questions (6% of questions) in Locating, but re-plan 77 out of 202 questions (38% of questions) in Building. In addition, the rate of questions that are re-planned more than four times is much larger in Building (30 questions, 14.85%) than Locating (7 ones, 1.75%). We note that an LM re-plans if its original plan is insufficient, as we explained in

Section 4. Thus, the result indicates that an LM struggles to make a plan (for both initial planning and re-planning) in Building and explains relatively small gap between PlanRAG-LM and other techniques in Building, in Table 4. It is also consistent with the result in Table 7 where the accuracy improved by re-planning decreases as the number of re-plannings increases.

6 Related Work

NLP tasks using structured data A number of benchmarks for reasoning over structured data have been proposed, such as Table NLI benchmarks (Chen et al., 2020a; Gupta et al., 2020; Jena et al., 2022) and Tabular QA (Iyyer et al., 2017; Chen et al., 2020b; Zhu et al., 2021; Chen et al., 2021; Wenhu Chen, 2021; Li et al., 2022; Nan et al., 2022). Tabular QA is a task answering questions based on given tabular data, and Tabular NLI is a task determining hypotheses are entailment, contradiction, or neutral based on given tabular data. However, such benchmarks did not consider the business rules, and also, not consider LMs' querying over a large structured database. Table 8 shows that the number of rows per table in the above benchmarks is much smaller than that in our DQA benchmark.

Benchmarks	Avg. rows
Tabular QA	
HybridQA (Chen et al., 2020b)	15.7
TAT-QA (Zhu et al., 2021)	9.5
FinQA (Chen et al., 2021)	6.26
WikiTableQA (Zhu et al., 2021)	30
Table NLI	
TabFact (Chen et al., 2020a)	14.5
ToTTo-TNLI (Jena et al., 2022)	35.8
Decision QA (ours)	
Building with RDB	579.0
Locating with RDB	**2038.8**

Table 8: Average number of rows per table in Tabular QA, Table NLI and Decision QA benchmarks. For Tabular QA and Table NLI, and DQA benchmarks. For DQA, the number of rows of D are presented since every question needs to access to all tables.

RAG technique RAG is the most common approach to augment the generation of LMs with external data. LMs retrieve the data related to an input (e.g., question) and then, generate a response (e.g., answer) based on the retrieved observations. Most of them operate in a single-turn (i.e., non-iterative) manner (Guu et al., 2020; Izac-ard et al., 2023; Izacard and Grave, 2021; Jiang et al., 2022b; Shi et al., 2023; Borgeaud et al., 2022; Lewis et al., 2020) and so have clear limitations in complex tasks that require multi-hop reasoning. To address this, several methods have been proposed to augment generation by performing a process of retrieval-then-generation iteratively (Jiang et al., 2023b; Shao et al., 2023; Trivedi et al., 2023; Jiang et al., 2023a). It has shown successful performance on various tasks that require multiple data accesses to generate responses (Yang et al., 2018; Thorne et al., 2018; Ho et al., 2020; Aly et al., 2021), but is not powerful enough to solve the decision making task well in our experiments.

7 Conclusions

In this paper, we explored the capability of LLMs as a solution for decision making. We proposed the new decision-making task, **Decision QA**, which answers the best decision for a given complex decision-making question that requires considering both the business rules and business situation represented in a large database (in either RDB or GDB). We built the benchmark for Decision QA, called **DQA**, by extracting 301 sets of a database (in both RDB and GDB), a question, and an answers(ground truth) from two popular video games imitating real business situations that require decision making. We also proposed the new RAG technique called **PlanRAG**, which performs planning before retrieving and re-planning if the initial plan is not good enough. Through extensive experiments, we demonstrated that PlanRAG significantly outperforms the SOTA iterative RAG for the Decision QA task.

8 Limitations

In this paper, we explored the capability of LLM as a solution for decision making. However, our study still has several limitations.

First, in this study, we focused on Decision QA using graph database or relational database. Decision making based on other databases, such as a hybrid form of database and vector database, could be explored in future research.

Next, we proposed techniques from a high-level RAG technique perspective that should be considered when solving Decision QA. Therefore, we do not focus on the low-level methods for solving Decision QA in this paper. For example, creating a fine-tuned model that efficiently generates Cypher

queries could be beneficial for solving Decision QA using GDB. These areas should also be addressed in future works.

Lastly, we designed our PlanRAG methodology and implemented our PlanRAG-LM using single LM, while some researches have suggest language model frameworks using multiple LMs. The effectiveness of the PlanRAG in multiple LMs framework is not a focus of this paper and is left as a further study.

9 Ethical Considerations

Language models have a hallucination issue and can potentially generate biased answers. RAG methods we have discussed are known to mitigate these issues to some extent, but it does not imply that these issues do not occur. Therefore, when applying our research to real-world applications, it is essential to closely examine whether the generated decisions are inferred based on hallucinated or biased knowledge.

Before constructing our benchmark and simulator from Europa Universalis IV and Victoria 3 games, we have considered end user license agreement (EULA)[6] of their game publisher, Paradox Interactive. Our benchmark and simulator correspond to gameplay and scripts of user generated content (UGC) in section 5 of EULA, and thus our content should be open-sourced. Therefore, we open our benchmark and simulator under the MIT license. Also, utilizing all icons that came from these games in our paper is classified as streaming Paradox Games in section 6 of EULA. According to EULA, we can freely use icons if our paper is not behind a paywall.

Video games that we have used to construct DQA describe historical situations. Therefore, our datasets, based on these games, include knowledge that contradicts contemporary common sense and might be aggressive towards certain groups. For example, the correct answer that a specific nation should influence a particular region in the Locating scenario of our benchmark might be aggressive to specific nations or regions. To avoid these issues, we anonymized the names of nations into three-letter codes rather than mentioning their names directly. For example, instead of using the term *Bahmanis Sultanate*[7], we employed the term *BAH*,

and instead of *The Papel States*[8], we used *PAP* as terminology.

Acknowledgements

References

Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29:626–688.

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. The fact extraction and VERification over unstructured and structured information (FEVEROUS) shared task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 1–13, Dominican Republic. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020a. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020b. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020,*

---

[6]https://legal.paradoxplaza.com/eula?locale=en

[7]https://en.wikipedia.org/wiki/Bahmani_Sultanate

---

[8]https://en.wikipedia.org/wiki/Papal_States

pages 1026–1036, Online. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mario José Diván. 2017. Data-driven decision making. In *2017 international conference on Infocom technologies and unmanned systems (trends and future directions)(ICTUS)*, pages 50–56. IEEE.

Sean Eom and E Kim. 2006. A survey of decision support system applications (1995–2001). *Journal of the Operational Research Society*, 57:1264–1278.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data*, pages 1433–1445.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568.

Vijay Gupta, Emmanuel Peters, Tan Miller, and Kelvin Blyden. 2002. Implementing a distribution-network decision-support system at pfizer/warner-lambert. *Interfaces*, 32(4):28–45.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Darius Hedgebeth. 2007. Data-driven decision making for the enterprise: an overview of business intelligence applications. *Vine*, 37(4):414–420.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

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. *Journal of Machine Learning Research*, 24(251):1–43.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*.

Aashna Jena, Vivek Gupta, Manish Shrivastava, and Julian Eisenschlos. 2022. Leveraging data recasting to enhance tabular reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4483–4496, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023a. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Zhengbao Jiang, Luyu Gao, Zhiruo Wang, Jun Araki, Haibo Ding, Jamie Callan, and Graham Neubig. 2022a. Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2336–2349, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhengbao Jiang, Luyu Gao, Zhiruo Wang, Jun Araki, Haibo Ding, Jamie Callan, and Graham Neubig. 2022b. Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2336–2349.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

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.

Fentahun Moges Kasie, Glen Bright, and Anthony Walker. 2017. Decision support systems in manufacturing: a survey and future trends. *Journal of Modelling in Management*, 12(3):432–454.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Moxin Li, Fuli Feng, Hanwang Zhang, Xiangnan He, Fengbin Zhu, and Tat-Seng Chua. 2022. Learning to imagine: Integrating counterfactual thinking in neural discrete reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 57–69, Dublin, Ireland. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

R OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

Daniel J Power. 2007. A brief history of decision support systems. *DSSResources. com*, 3.

Daniel J Power. 2008. Understanding data-driven decision support systems. *Information Systems Management*, 25(2):149–154.

Foster Provost and Tom Fawcett. 2013. Data science and its relationship to big data and data-driven decision making. *Big data*, 1(1):51–59.

Roberto Sala, Fabiana Pirola, Giuditta Pezzotta, and Sergio Cavalieri. 2022. Data-driven decision making in maintenance service delivery process: A case study. *Applied Sciences*, 12(15):7395.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *arXiv preprint arXiv:2305.15294*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Orlando Troisi, Gennaro Maione, Mara Grimaldi, and Francesca Loia. 2020. Growth hacking: Insights on data-driven decision-making from three firms. *Industrial Marketing Management*, 90:538–557.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Eva Schlinger William Wang William Cohen Wenhu Chen, Ming-wei Chang. 2021. Open question answering over tables and text. *Proceedings of ICLR 2021*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

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 *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, Richard James, Jure Leskovec, Percy Liang, Mike Lewis, Luke Zettlemoyer, and Wen-Tau Yih. 2023. Retrieval-augmented multimodal language modeling. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 39755–39769. PMLR.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Appendix

A.1 Data collection

To collect D, we first select the provided savefile from each game and extract related data by game savefile parser. The extracted data was stored in D according to the DB schema of each scenario. Next, we generate Q by applying components of D into pre-defined question format for each scenario. To control the quality of our benchmark, we consider the following points:

- For the Locating scenario, we create one problem for each country. As *profit* is determined by $TP_{country}$, we omit countries with low $TP_{country}$ due to their minimal impact on decisions.

- For the Building scenario, we filtered problems which have multiple answers.

A.2 Data instances

For both scenarios, the instances share the same business rules R. Some instances share the same database D, but not all use the same one. Here, each database D has a different number of rows and different column values (e.g., LV, size of the building, current price of goods).

In the Locating scenario, every instance has a different goal with a different pair of {COUNTRY_NODE} and {HOME_NODE} in the template of questions of *"Where should I locate my merchant? My goal is maximizing* {COUNTRY_NODE}*'s profit on home node,* {HOME_NODE}*"*.

In the Building scenario, every instance has a different {GOODS} in the template of questions of *"Which building ID should we increase a level by 5 to maximally decrease the market price of* {GOODS}*?"*.

The instances in both scenarios are designed that LMs should retrieve different intermediate nodes and edges to calculate the values, such as *TPR* for Locating and *TS* for Building, required to make a best decision.

A.3 Simulators for annotation

Although applying every decision to real games and comparing the results is the most credible approach to annotate the best decision, it is impossible due to the following characteristics of games: (1) randomness and (2) not being open-sourced. Thus, we

develop simulators for each scenario on DQA to calculate decision results deterministically.

Our simulator takes D with *decision* as input, calculate effects of decision, and returns updated D by simulating based on business rules. To annotate answers for a question, we simulates all decision candidates. For example, in Figure 1, we locates a merchant on Doab and Ganges, our simulator calculates profit increments, 0.23 and 0.03 for each decisions, respectively. From these calculated results, we can annotate that Doab is the best decision for the question. Detailed algorithms are in Appendix A.4

A.4 Algorithm of simulators

We provide an algorithm of simulator for each scenario for future research. Before explaining each simulator, we introduce some additional symbols in Table 9, while other symbols are described in Section 3.

Symbol	Description
Locating	
obj_n	The node we should calculate its flow
Nodes	The set of nodes
Countries	The set of countries
NodeCountry	The set of NodeCountry relationships
Simulated_n	The set of nodes which are simulated
Building	
Goods	The set of goods
Buildings	The set of buildings
cycle_cnt	# of hops that a building affects
$MIN(x, y)$	Minimum value between x and y
$AVG_Y(X)$	For $y \in Y$, An average of the set $X(y)$

Table 9: Summary of additional symbols.

A.4.1 Locating simulator algorithm

Algorithm 1 describes the mechanism of Locating simulator. There are three different functions in the Locating simulator: 'trading_power_estimate', 'flow_estimation', and 'find_top_n'.

- **trading_power_estimate** takes the database D as an input, calculates and updates $TP_{country}(n, c)$ and $TP_{total}(n)$ for all $n, c \in NodeCountry$, and returns updated D.

- **flow_estimation** takes the database D and an objective node *obj_n* as inputs, calculates and updates *flow(obj_n, dest)* and IV(*dest*) for all *dest* such that *(obj_n, dest)* $\in F$ by using Eq. (2), and returns updated D.

- **find_top_n** takes the database D and *Simulated_n* as inputs and returns a node $n \in Nodes - Simulated\_n$ such that $\forall_{n' \in Nodes\text{-}Simulated\_n}(n, n') \notin F$. If $Nodes - Simulated\_n = \emptyset$, it returns *NULL*.

Algorithm 1: Locating simulator

Input: D /* Database for Locating */
Output: D /* Database after calculating a *profit* for every country in D. */
1   $D \leftarrow$ trading_power_estimate(D) /* Initialize $TP_{country}(n, c)$ and $TP_{total}(n)$ */
2   *Simulated_n* $\leftarrow$ { }   /* empty set. */
3   **do**
4   *obj_n* $\leftarrow$ find_top_n(*D, Simulated_n*)
5      $D \leftarrow$ flow_estimation(*D, obj_n*)
6   *Simulated_n*.add(*obj_n*)
7   **while** $Nodes - Simulated\_n \neq \emptyset$;
8   **for** $c, h \in Countries$ **do**
9      $profit(c) \leftarrow (LV(h, c) + IV(h, c)) \cdot \frac{TP_{country}(h, c)}{TP_{total}(h)}$
        /* Eq. (1, 3) */
10   **return** D

A.4.2 Building simulator algorithm

We provide the mechanism of Locating simulator in Algorithm 2. We initialize $TS(g)$ as the sum of $MS(g, b)$ to avoid the situation where the price of the goods falls to the local minimum. We also limit maximum *cycle_cnt* as 10 to make our simulator more efficient.

A.5 Business rules in DQA

Figure 7 shows the business rules R used as LM's input in Locating and Building. These are textual descriptions of the contents in Section 3.

A.6 Prompts for DQA

A.6.1 Previous RAG-based LMs

Figure 8 and 9 show an example of the prompts used for the single-turn RAG-based decision maker, SingleRAG-LM, in the Locating scenario with GDB setting. Figure 10 shows an example of the prompt used for the iterative RAG-based decision maker, IterRAG-LM, in the Locating scenario with GDB setting. Those prompts are based on the structure of ReAct, 'Thought'-'Action'-'Observation'.

A.6.2 PlanRAG-based LM

The prompt used to implement our PlanRAG-based decision maker, PlanRAG-LM, has a structure of 'Plan'-'Thought'-'Action'-'Observation'-'Re-plan'. Figure 11 shows an example of the prompt for Locating scenario with GDB setting.

Algorithm 2: Building simulator

Input: D /* Database for Building */
Output: D /* Database after calculating prices*/

1   **for** $g \in Goods$ **do**
2     $TD(g) \leftarrow PD(g)$ /* Eq. (4)*/
3     $TS(g) \leftarrow 0$ /* Eq. (5)*/
4   **for** $g, b \in Dem$ **do**
5     $TD(g) \leftarrow TD(g) + MD(g, b)$ /* Eq. (4)*/
6   **for** $g, b \in Sup$ **do**
7     $TS(g) \leftarrow TS(g) + MS(g, b)$ /*TS initialize*/
8   $cycle\_cnt \leftarrow 0$
9   **while** $cycle\_cnt < 10$ **do**
10   **for** $g, b \in Dem$ **do**
11      $CI(g, b) \leftarrow MD(g, b) * \frac{TS(g)}{TD(g)}$
       /* From business rules in Figure 7 (b). */
12   **for** $g \in Goods$ **do**
13      $TS(g) \leftarrow 0$ /* Eq. (5)*/
14   **for** $g, b \in Sup$ **do**
15      $CO(g, b) \leftarrow MS(g, b) \cdot$
       $AVG_{\{b'|(g,b')\in Dem\}}(MIN(1, \frac{CI(g,b')}{MD(g,b')}))$
       /* From business rules in Figure 7 (b). */
16   **for** $g, b \in Sup$ **do**
17      $TS(g) \leftarrow TS(g) + CO(g, b)$ /* Eq. (5)*/
18    $cycle\_cnt \leftarrow cycle\_cnt + 1$
19   **for** $g \in Goods$ **do**
20    $CP(g) \leftarrow BP(g) \cdot (1 + 0.75 \cdot \frac{TD(g)-TS(g)}{max(TD(g),TS(g))})$
     /* Eq. (6)*/
21   **return** D

The prompt structure was designed empirically through experiments conducted on both DQA scenarios.

In the experiments, we also considered the following two variations of the prompt structure: (1) Act without additional reasoning about a current step after planning (i.e., 'Plan'-'Action'-'Observation'-'Re-plan'), (2) Reason about planning in advance (i.e., 'Thought'-'Plan'-'Action'-'Observation'-'Re-plan'). The experiments were conducted by the same setup as in Section 5.1, using GPT-4 with zero temperature as base LMs in a zero-shot setting.

Table 10 shows the experimental results of the prompt structure of PlanRAG-LM, two aforementioned variations of that, and the ReAct prompt structure of IterRAG for 10% of questions in each DQA scenario. Here, in each scenario, we sampled the questions randomly at 10% from each DB setting. In the results, the prompt structure of PlanRAG-LM outperforms other baselines in both DQA scenarios. Meanwhile, one PlanRAG-LM variation using the prompt structure of 'Thought'-'Plan'-'Action'-'Observation'-'Re-plan' fails to demonstrate its effectiveness. An-

(a) A "Trading node" has a "local_value", "total_power", "outgoing", "ingoing" and **whether it's inland**. A "Country" has a "name", "development" and a "home_node" (home node). Between "Trading nodes", there can be a directional edge [source]. It connects from a higher node to a lower node. A "Country" can have a non-directional connection to a trading node. Each connection has a unique "base_trading_power" for each node. If a specific node is the home node of a country, that country earns profit from that node. The profit is proportional to the "local_value" plus "ingoing" and the ratio of the country's trading power and the total trading power of that node. i.e. (local_value+ingoing)*(country_trading_power / total_trading_power)
If a specific node is source of a country's home node, the country moves a value to dest node, proportional to the ratio of the country's trading power to the node's total trading power and (local_value + ingoing). In the dest node, the moved value is increased by 1.05 times and added to the ingoing. A "Merchant" belongs to a country and can be assigned to a specific trading node. A "Merchant" belonging to a trading node adds 2 to the trading power of the country's trading node and amplifies it by 1.05 times. *If one of the edges has an inland node, the added value changes to 2+max(development/3, 50). (optional)* If a specific trading node has more than one dest node, and a country that doesn't have that node as its home node places a "Merchant" on the Trading node, it can decide which dest node to move the "current_value" to. That is, the country can move a "current_value" proportional to its trading power to a specific dest node. If no merchant is placed, when there's more than one dest node, they lose the right to decide the direction. In other words, the country's "current_value" proportional to its trading power flows out in proportion to other outflows to the dest nodes. If there's only one dest node, it doesn't matter.

(b) Goods have a "name", corresponding "code", "base_price", "current_price", and "pop_demand". Buildings have a unique "id" and a "name" and "level" corresponding to their type. There exists a relation called Supply from Buildings to Goods. Supply has "max_supply", "current_output", and "level". The level here is the same as the level of the Building. Furthermore, max_supply and level have a proportional relationship. There exists a relation called Demand from Goods to Buildings. Demand has "max_demand", "current_input", and "level". Also, max_demand and level have a proportional relationship. The demand of Goods is defined as the Goods' "pop_demand" plus sum of the "max_demand" of all Demands connected to the Goods. The supply of Goods is defined by the sum of "current_output" of all Supplies connected to Goods. The "current_input" of Demand is determined by the ratio of connected Building's "max_demand" to connected Goods' demand, and multiplied by the supply of Goods. The "current_output" of Supply is determined by the average ratio of the connected Building's "current_input" to connected Goods' "max_demand", and multiplied by the "max_supply" of Supply. The "current_price" of Goods is determined by base_price*(1+0.75(demand-supply)/max(demand,supply)).

Figure 7: Business rules for (a) Locating and (b) Building.

other variation of PlanRAG-LM shows slightly better performance than the ReAct structure. The difference between these two results comes from

```
# Prefix
You are a decision-making agent answering a given question.
You should collect the data to answer the question:
# Tool descriptions
Graph DB: Useful for when you need to collect the data that
follows the following schema (You MUST generate a Cypher query
statement to interact with this tool):
(n:Trade_node {{name, local_value, is_inland, total_power,
outgoing, ingoing}});
(m:Country {{name, home_node, development}});

(Trade_node)-[r:source {{flow}}]->[Trade_node]
(Country)-[NodeCountry{{is_home,
has_merchant,base_trading_power,calculated_trading_power}}]-
>(Trade_node), args: {{{{'tool_input': {{{{'type': 'string'}}}}}}}}
Self thinking: Useful for when there is no available tool., args:
{{{{'tool_input': {{{{'type': 'string'}}}}}}}}
# Format instructions
Use the following Strict format:
Question: the input question you must answer.
Thought: you should always think about what to do.
Action: a suitable database name, MUST be one of ['Graph DB',
'Self-thinking'].
Action input: a syntactically correct query statement only, MUST
be written by Cypher query language.
Observation: the result of the action.
Thought: I now know the answer.
Final answer: the final answer to the question based on the
observed data.
# Suffix
Begin! Keep in mind that Your response MUST follow the valid
format above.
```

Figure 8: Retrieval prompt for a single-turn RAG technique on GDB case in Locating scenario.

```
# Prefix
You are a decision-making agent answering a given question.
You have already collected the data to answer the question.
Indeed, you should make your Final answer immediately.:
# Tool descriptions
Graph DB: Useful for when you need to collect the data that
follows the following schema (You MUST generate a Cypher query
statement to interact with this tool):
(n:Trade_node {{name, local_value, is_inland, total_power,
outgoing, ingoing}});
(m:Country {{name, home_node, development}});

(Trade_node)-[r:source {{flow}}]->[Trade_node]
(Country)-[NodeCountry{{is_home,
has_merchant,base_trading_power,calculated_trading_power}}]-
>(Trade_node), args: {{{{'tool_input': {{{{'type': 'string'}}}}}}}}
Self thinking: Useful for when there is no available tool., args:
{{{{'tool_input': {{{{'type': 'string'}}}}}}}}
# Format instructions
Use the following Strict format:

Final answer: the final answer to the question based on the
observed data.
# Suffix
Begin!
```

Figure 9: Answer generation prompt for a single-turn
RAG technique on GDB case in Locating scenario.

```
# Prefix
You are a decision-making agent answering a given question.
You should collect the data to answer the question.
Keep in mind that the question can require to access following
databases multiple times:
# Tool descriptions
Graph DB: Useful for when you need to collect the data that
follows the following schema (You MUST generate a Cypher query
statement to interact with this tool):
(n:Trade_node {{name, local_value, is_inland, total_power,
outgoing, ingoing}});
(m:Country {{name, home_node, development}});

(Trade_node)-[r:source {{flow}}]->[Trade_node]
(Country)-[NodeCountry{{is_home,
has_merchant,base_trading_power,calculated_trading_power}}]-
>(Trade_node), args: {{{{'tool_input': {{{{'type': 'string'}}}}}}}}
Self thinking: Useful for when there is no available tool., args:
{{{{'tool_input': {{{{'type': 'string'}}}}}}}}
# Format instructions
Use the following Strict format:
Question: the input question you must answer.
Thought: you should always think about what to do.
Action: a suitable database name, MUST be one of ['Graph DB',
'Self-thinking'].
Action input: a syntactically correct query statement only, MUST
be written by Cypher query language.
Observation: the result of the action.
... (a process of Thought, Action, Action input, and Observation can
repeat together N times)
Thought: I now know the answer.
Final answer: the final answer to the question based on the
observed data.
# Suffix
Begin! Keep in mind that Your response MUST follow the valid
format above.
```

Figure 10: Prompt for an iterative RAG technique on
GDB case in Locating scenario.

cess 'Thought' have been well discussed in several
studies (Yao et al., 2023; Wei et al., 2022). There-
fore, these results can show the significance of the
planning process in decision-making tasks such as
Decision QA.

Prompt structure	Locating	Building
IterRAG-LM		
Thou-Act-Obs	37.5	30
PlanRAG-LM		
Plan-Thou-Act-Obs-Replan	**57.5**	**50**
PlanRAG-LM variations		
Plan-Act-Obs-Replan	40	40
Thou-Plan-Act-Obs-Replan	27.5	30

Table 10: Accuracy(%) of LMs for DQA by prompt
structure. 'Thou', 'Act', and 'Obs' means 'Thought',
'Action', and 'Observation', respectively.

A.7 Experimenting with other models

In this section, we implement IterRAG-LM and
PlanRAG-LM by four different models: (1) GPT-

whether to reason about the next action to take
at each iteration, 'Thought', or to act based on
the plan established through the planning process,
'Plan'. The importance and effectiveness of the pro-

```
# Prefix
You are a decision-making agent answering a given question.
You should collect the data to answer the question.
To this end, firstly, you need to plan which data would be needed
in what order.
Keep in mind that the question can require to access following
databases multiple times:
# Tool descriptions
Graph DB: Useful for when you need to collect the data that
follows the following schema (You MUST generate a Cypher
query statement to interact with this tool):
(n:Trade_node {{name, local_value, is_inland, total_power,
outgoing, ingoing}});
(m:Country {{name, home_node, development}});

(Trade_node)-[r:source {{flow}}]->[Trade_node]
(Country)-[NodeCountry{{is_home,
has_merchant,base_trading_power,calculated_trading_power}}]-
>(Trade_node), args: {{{{'tool_input': {{{{'type': 'string'}}}}}}}}
Self thinking: Useful for when there is no available tool., args:
{{{{'tool_input': {{{{'type': 'string'}}}}}}}}
# Format instructions
Use the following Strict format:
Question: the input question you must answer.
Plan: [Step 1: requirement 1, Step 2: requirement 2, ..., Step N:
requirement N].
Current step: the current Step in the Plan.
Thought: you should always think about the Current step.
Action: a suitable database name, MUST be one of ['Graph DB',
'Self-thinking'].
Action input: a syntactically correct query statement only, MUST
be written by Cypher query language.
Observation: the data from the database.
Re-plan: respond with 'Y' and change your Plan if you think a
current Plan is not helpful, otherwise respond with 'N' and
continue a process based on the current Plan.
... (a process of Plan, Current step, Thought, Action, Action input,
Observation, and Re-plan can repeat N times)
Thought: I now know the answer.
Final answer: the final answer to the question based on the
observed data.
# Suffix
Begin! Keep in mind that Your response MUST follow the valid
format above.
```

Figure 11: Prompt for PlanRAG technique on GDB case
in Locating scenario.

3.5[9] (2) Llama 2 (70B), (3) Llama 2 (13B) (Tou-
vron et al., 2023), and (4) Phi-2 (Javaheripi et al.,
2023). All experiments are conducted on a single
machine equipped with eight Nvidia A100 (80GB)
GPUs. To accelerate inference speed, we utilize
vLLM (Kwon et al., 2023) library for open models
inference. We set temperature to zero and 0.1 for
GPT-3.5-turbo and other open models, respectively.
Other settings are consistent with those described in
Section 5. We provide results of PlanRAG-LM and
IterRAG-LM by four models in Table 11. In the
result, PlanRAG-LM and IterRAG-LM by Llama-2
and Phi-2 models cannot solve any problems in
DQA. By GPT-3.5, IterRAG-LM shows batter per-
formance rather than PlanRAG-LM. It is because
the prompt of PlanRAG is too complex for GPT-

---

[9]gpt-3.5-turbo-0125, which is the latest gpt-3.5 model.

3.5 to understand instructions and generate proper
answers.

Models	Locating		Building	
	RDB	GDB	RDB	GDB
GPT-3.5				
IterRAG-LM	**8.0**	2.5	**22.8**	**3.96**
PlanRAG-LM	0	**4.0**	1.0	1.0
Llama 2 (70B)				
IterRAG	0	0	0	0
PlanRAG	0	0	0	0
Llama 2 (13B)				
IterRAG	0	0	0	0
PlanRAG	0	0	0	0
Phi-2				
IterRAG	0	0	0	0
PlanRAG	0	0	0	0

Table 11: Accuracy(%) of IterRAG-LM and PlanRAG-
LM using several models.

A.8 Re-planning cases

Table 12 presents statistics and examples of re-
plannings conducted by PlanRAG-LM. We cat-
egorized all re-planning cases into three groups:
(1) Increase, **(2) Same**, and **(3) Decrease**, where
"Increase" means the number of steps increases
after re-planning compared to the original plan,
"Same" means the number of steps is the same with
the number of steps of the original plan, and "De-
crease" means the number of steps decreases after
re-planning.

Each category is further divided into the follow-
ing sub-categories:

- **Re-order** includes cases where a sequence of
  steps is arranged.
- **Replace** includes cases where some steps are
  substituted with new steps.
- **Change target** includes cases where targets
  of actions, such as lookup or calculation, are
  changed.
- **Add look-up** includes cases where new
  lookup actions are added to an original plan.
- **Add calculation** includes cases where new
  calculation actions are added to an original
  plan.
- **Add both actions** includes cases where both
  the lookup and the calculation actions are
  added to an original plan by a single re-
  planning process.
- **Divide to sub-steps** includes cases where a
  single step of an original plan is divided into

sub-steps by breaking it down into detailed actions.

- **Delete** includes cases where some steps are removed from an original plan.
- **Merge** includes cases where some steps are summarized or merged into a single step.

As shown in Table 7, since re-planning is done more than twice for some questions, they can belong to several categories. In the Locating scenario, PlanRAG-LM primarily performs re-planning of "Add look-up". In the case of Building, it is observed that "Divide to sub-steps" is the predominant strategy for re-planning across most cases.

Category		Example		Occurrence rate	
		Original plan	New plan	Locating	Building
Same	Re-order	…, **Step 2: Calculate the potential profit increase for each of these nodes if a merchant is placed there, Step 3: Identify the node with the development of the country "MJZ"**, …	…, **Step 2: Identify the node with the development of the country "MJZ", Step 3: Calculate the potential profit increase for each of these nodes if a merchant is placed there,** …	16.67%	0.00%
	Replace	…, **Step 6: Calculate the sum of max_demand from demand table where goods_id is 'oil'**, …	…, **Step 6: If there is no max_demand for 'oil' in the demand table, set max_demand to 0,** …	16.67%	44.44%
	Change target	…, Step 3: Calculate **the current demand** for groceries **from the demand table**, …	…, Step 3: Calculate **the pop_demand** for groceries **from the goods table**, …	25.00%	34.92%
Increase	Add look-up	Step 1: Identify the trading nodes that are upstream of "yumen", Step 2: Calculate the potential profit increase for each of these nodes if a merchant is placed there, …	Step 1: Identify the trading nodes that are upstream of "yumen", **Step 2: Check if the source nodes are inland or not,** Step 3: Calculate the potential profit increase for each of these nodes if a merchant is placed there, …	66.67%	41.27%
	Add calculation	…, Step 4: Check if the nodes are inland or not, as this will affect the added value of the merchant.	…, Step 4: Check if the nodes are inland or not, as this will affect the added value of the merchant. **Step 5: If the nodes are inland, calculate the potential profit considering the development of the country.**	16.67%	38.10%
	Add both actions	Step 2: Calculate the potential profit increase for each of these nodes if a merchant is placed there, Step 3: Compare the potential profit increases and identify the node with the highest potential profit increase.	Step 2: Calculate the potential profit increase for each of these nodes if a merchant is placed there, **Step 3: Check if the nodes are inland, Step 4: Calculate the merchant power based on whether the node is inland,** Step 5: Compare the potential profit increases and identify the node with the highest potential profit increase.	8.33%	4.76%
	Divide to sub-steps	…, Step 2: Retrieve all building ids that supply groceries from the supply table, **Step 3: Calculate the current demand and supply for groceries,** …	…, Step 2: Retrieve all building ids that supply groceries from the supply table, **Step 3: Calculate the current demand for groceries from the demand table, Step 4: Calculate the current supply for groceries from the supply table,**…	8.33%	77.78%
	Others	…, Step 4: Retrieve the relationship between building level and supply of goods, Step 5: Calculate the new price of "tools" if the level of each building is increased, ...	…, Step 4: Retrieve the relationship between building level and supply of goods, **Step 5: Assume a linear relationship between building level and supply of goods,** Step 6: Calculate the new price of "tools" if the level of each building is increased, ...	4.17%	6.35%
Decrease	Delete	…, Step 2: Retrieve all buildings that supply steel, **Step 3: Retrieve all buildings that demand steel,** Step 4: Calculate the potential decrease in the market price of steel for each building if the level is increased by 5, …	…, Step 2: Retrieve all buildings that supply steel, Step 3: Calculate the potential decrease in the market price of steel for each building if the level is increased by 5, …	4.17%	12.70%
	Merge	…, Step 2: Identify the trading nodes where "MCH" has a presence, **Step 3: Calculate the potential profit increase for each of these nodes if a merchant is placed there, Step 4: Identify the node with the highest potential profit increase.**	…, Step 2: Identify the trading nodes where "MCH" has a presence, **Step 3: Since "MCH" only has a presence in "girin", the merchant can only be placed there.**	4.17%	0.00%

Table 12: Category-wise statistics and examples of re-planning cases. The occurrence rate indicates the proportion of re-planning cases within a specific category relative to the total number of questions that re-plannings are conducted.