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

**Anonymous ACL submission**

## 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 given natural language question $Q$ and data $D$.There is no benchmark which can examine Decision QA, we propose Decision QA benchmark, **DQA**, composed of the locating and building scenarios constructed from two video games (Europa Universalis IV and Victoria 3) which 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**). In our PlanRAG, the PlanRAG-based LM generates the plan for data analysis in the first planning step, and the retriever generates the queries for data analysis in the second retrieving step. The proposed method outperforms the state-of-the-art iterative RAG method by 12.4% in the locating scenario and by 1.8% in the building scenario, respectively.

## 1 Introduction

Decision making is the process of exploring multiple alternatives to achieve a specific goal, collecting and analyzing data, and then selecting one of the alternatives based on the data analysis (Provost and Fawcett, 2013; Diván, 2017). For example, determining supply on a company by analyzing the market or managing resources, precise decision making plays a crucial role in the success of the company (Kasie et al., 2017). To make the best decision, it is necessary to analyze extensive and diverse data. Since this process is challenging, a lot of decision support systems have been researched to make it easier (Eom and Kim, 2006; Power, 2007; Hedgebeth, 2007; Power, 2008; Kasie et al., 2017). However, determining which data analysis is needed before analyzing data itself remains a human role, thus decision making remains a complex and challenging problem.

Recently, Large Language Models (LLMs) pretrained on vast corpora have demonstrated remarkable versatility across a wide range of natural language tasks (Brown et al., 2020; OpenAI, 2023). Consequently, some researchers have tried to integrate LLMs with external data and utilize them (Jiang et al., 2023a; Patil et al., 2023). Despite these efforts, research on utilizing LLMs as an end-to-end decision-making solution is rare, because of a lack of task definition, effective methods for the task, and the benchmark for evaluating the decision-making capabilities of LLMs.

To address these issues, we first 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 data $D$ and a natural language 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 to locate a merchant to a specific trading city (post) in order to maximize its profit on its main trading post(home). The example shows that a decision-making LLM decides to locate a merchant in Doab to maximize the profit of Deccan, the home trading post of the country BAH, after analyzing the data about the state of international trade.

Next, we propose a benchmark for Decision QA called **DQA**. Due to the difficulty in verifying real-world decision-making outcomes, we generate datasets and questions of our benchmark by adopting game systems from two video games that require decision making: Europa Universalis IV and Victoria 3[1]. To eliminate the randomness of the game and publish our benchmark, we also develop game simulators that the decision outcome

[1]Grand strategy games published by Paradox Interactive

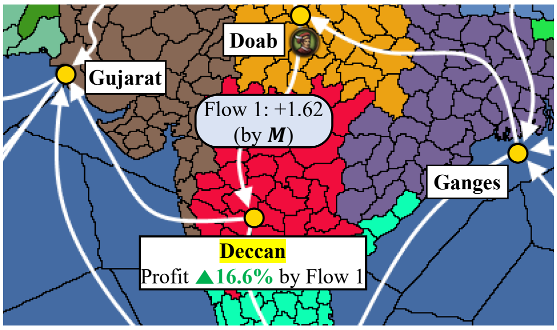
**Step 1: Data analysis for given input data and question**

Where should I locate my merchant to steer trade to **Deccan**? Note that my goal is maximizing **BAH**'s profit on **Deccan**.

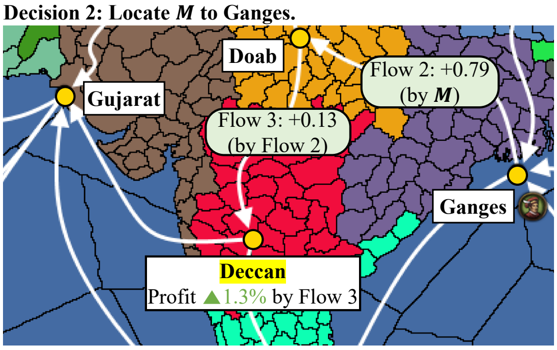
| trade post | local value | incoming value | $TP_{total}$ | $TP_{BAH}$ | $TP_{DLH}$ | ... |
|---|---|---|---|---|---|---|
| **Deccan** | 8.91 | 0.83 | 1128 | 186 | 0 | ... |
| **Doab** | 6.98 | 0.87 | 1243 | 71 | 53.2 | ... |
| **Ganges** | 8.31 | 1.07 | 1172 | 0 | 5.33 | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | |

| upstream | downstream | flow | ... |
|---|---|---|---|
| Doab | Deccan | 0.78 | ... |
| Ganges | Doab | 0.82 | ... |
| ⋮ | ⋮ | ⋮ | |

**Decision 1: Locate *M* to Doab.**




**Decision 2: Locate *M* to Ganges.**




**Step 2: Answering based on the result of data analysis**

You should locate your merchant to the **Doab** trading node to steer trade to Deccan and maximize your profit.

Figure 1: The example of Decision QA. A yellow dot on the map represents each trading node. A "Profit" in a box indicates potential profit change by each decision. Note that the potential profit increases are not explicitly mentioned in the provided data. $M$, $TP_T$, $TP_{BAH}$ and $TP_{DLH}$ mean a merchant, the total trading power, the trading power for BAH and the trading power of DLH respectively.

for each scenario of the games. We utilize these simulators as annotators for the questions of DQA.

For Decision QA, LLMs usually require to follow these two steps: (a) data analysis for given input data and question, and (b) answering based on the result of data analysis. In Step (a), LLMs are necessary to access the data that has not been used during pre-training, namely external data. For accessing external data and answering based on it, a lot of methods based on the Retrieval-Augmented Generation (RAG) technique have been proposed(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 this technique, a retriever finds external data that is highly relevant to a question and conveys it to LMs, so that LMs can generate an answer based on the retrieved data (Lewis et al., 2020). Recently, the iterative RAG technique has also been proposed to address more complex problems which utilizes retrieved results to perform further retrievals (Trivedi et al., 2023; Jiang et al., 2023b). The language models based on these RAG techniques have shown significant improvement in knowledge-intensive tasks such as open-domain QA (Karpukhin et al., 2020) and open-domain conversation (Xu et al., 2022). However, just retrieving relevant facts is not enough to solve Decision QA. It is necessary to understand the problem to determine what data analysis needs to be performed and then retrieve necessary data. Therefore, the previous RAG techniques tend to show a weakness in solving Decision QA.

As a new RAG technique, we propose the iterative plan-then-retrieval augmented generation technique, **PlanRAG**, which is extended from the iterative RAG technique for Decision QA. In this technique, an LM first generates retrieval plan for data analysis by examining data schema and questions (the planning step). Next, the LM generates data-retrieving queries according to the plan and executes them to external data the retrieving step. After the retrieval, the LM assesses whether it needs to make new plan for further retrieval, and does re-planning if necessary.

To validate the effectiveness of PlanRAG on Decision QA, we applied both the state-of-the-art iterative RAG-based LM and the PlanRAG-based LM to the DQA benchmark, and showed that PlanRAG is far more effective for Decision QA. Our contributions are summarized as follows:

- We define a new challenging task, **Decision QA**, which requires data analysis to make the best decision.
- We propose the benchmark for Decision QA called **DQA**.

- We propose a new retrieval-augmented generation technique, **PlanRAG**, which enhances decision-making capabilities of LLMs.
- We demonstrated that our PlanRAG significantly outperforms the state-of-the-art retrieval-augmenting techniques for Decision QA task.

## 2 Related Work

Retrieval-then-generation is the most commonly used approach to augment the generation capabilities of generative LMs with external data. Retrieval-augmented LMs retrieve 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 and use a dense vector similarity search method as a retriever (Guu et al., 2020; Izacard et al., 2023; Izacard and Grave, 2021; Jiang et al., 2022b; Shi et al., 2023; Borgeaud et al., 2022; Lewis et al., 2020). This single-turn approach has clear limitations in complex tasks that require multi-hop reasoning due to the partial nature of relevant data.

To address this, several methods have been recently proposed to augment the final response generation of generative LMs by iteratively performing a process of retrieval-then-generation (Jiang et al., 2023b; Shao et al., 2023; Trivedi et al., 2023; Jiang et al., 2023a). In this iterative retrieval-then-generation approach, the role of generative LMs is extended from response generation for input to intermediate query generation for retrieval. In this approach, an LM (Language Model) performs the retrieval process again, based on the queries it generates. This approach has shown successful performance on various tasks that require external data to generate responses (Yang et al., 2018; Thorne et al., 2018; Ho et al., 2020; Aly et al., 2021).

## 3 Problem Definition

We define **Decision QA** as the task of answering the best decision $d_{best}$ by understanding a given natural language question $Q$ and analyzing given data $D$. Here, $Q$ contains a textual goal that requires a specific decision to achieve it. The best decision $d_{best}$ should meet the goal presented in $Q$ and can be inferred by appropriately analyzing $D$.

In general, the data $D$ in Decision QA is too large to fit in as an input of an LM. Therefore, we assume that an LM retrieves data from $D$ for their analysis of Decision QA. In this paper, we consider Labeled Property Graph (LPG) for the format of $D$ to representthe relationships among entities (e.g., trading posts) as edges and the attributes of entities (e.g., local value) as vertices (Akoglu et al., 2015; Guo et al., 2020).

Decision QA has two different characteristics that distinguish it from the existing QA tasks: (1) The best decision in Decision QA is not explicitly mentioned in the provided data. Thus, an LM must infer it through data analysis, while the facts in the existing QA tasks such as open-domain QA (Joshi et al., 2017) and KGQA (Yang et al., 2018) can be retrieved explicitly from given data. For example, an LM should calculate the potential profit of nodes (i.e., trading posts) and infer the best location of a merchant (e.g., Doab). (2) The questions in Decision QA do not provide any data analysis method, while the existing QA tasks such as Tabular QA (Zhu et al., 2021; Li et al., 2022) provides the required data analysis method explicitly. Thus, an LM should determine the method itself. For example, in Figure 1, an LM should try to identify the neighbor nodes of a given node, Deccan, even though there is no such a hint in the question.

## 4 DQA: Decision QA benchmark

### 4.1 Backgrounds

The DQA benchmark is constructed by two different game scenarios: (1) **Locating scenario** from the Europa Universalis IV game, (2) **Building scenario**, from the Victoria 3 game.

**Locating scenario:** We first explain the overview of the locating scenario using Figure 1. Here, $Q$ asks for the best merchant location where the country named BAH can maximize its profit on Deccan. $D$ is composed of the following components:

- A set of trading nodes, each of which has its own local value(LV) and incoming value(IV), and the total trading power ($TP_{total}$).
- A set of upstream-downstream relationships between two trading nodes.
- A set of countries each of which has its own trading power (or amount of influence) on each trading node. The total trading power of a trading node is the sum of all trading powers of all countries on the trading node. Each country has a single specific trading node as its main trading node (home node).

Before we explain the scenario, we define following three kinds of intermediate values: (1) $OV$: the overall value on a node, (2) $TPR$: the ratio of trading power of a country on a node to the total trading power of the node, and (3) $CV$: the amount of the value that is controlled by a country on a node. First, $OV$ is defined as $OV = IV + LV$. For example, in Deccan in Figure 1, $OV$ is calculated as $OV = 8.91 + 0.83 = 9.74$. Second, $TPR$ is defined as $TPR = TP_{country}/TP_{total}$. In Figure 1, $TPR$ of BAH on Deccan is calculated as $186/1128 \approx 0.165$. Third, $CV$ is defined as $CV = OV * TPR$ for a pair of a country and a node. In Figure 1, $CV$ of BAH in Deccan is $0.165 * 9.74 \approx 1.61$. The profit of a country is defined as $CV$ of the country on its home node. Thus, the profit of BAH is 1.61. For the non-home nodes of a country,, the country transfers $CV$ from them to its downstream node(s). We denote the amount of transfer as *flow*. For example BAH on Doab in Figure 1, the flow by BAH is calculated as $(6.98 + 0.87) * 71/1243 \approx 0.45$. Here, the incoming value of the downstream node is defined as the sum of all the flows from its upstream nodes.

In this scenario, a merchant increases the flow toward the home node. Thus, To calculate a profit increment, an LM needs to: (1) Determine the nodes where the merchant can be positioned by examining the upstream nodes of the home node. In Figure 1, Doab and Ganges are examples of these. (2) Ascertain how much the merchant increases the flow. In Figure 1, the flow from Doab to Deccan is increased by 1.62 due to Decision 1, and the flow from Ganges to Doab and from Doab to Deccan is increased by 0.79 and 0.13, respectively, due to Decision 2. (3) evaluate the increment in the overall value which resulting from these decisions. In Figure 1, the overall value of Deccan is increased by 1.62 (+16.6%), due to $d_1$, and by 0.13 (+1.3%), due to Decision 2. As we mentioned, these are proportional to the profit. Thus, we can determine "Decision 1: Locating merchant to Doab" as the $d_{best}$ for this example.

We next explain how a merchant can affect to the specific flow by Figure 2. The table in Figure 2 provides the local value and $TPR$[2] for each node. For simplicity, we assume that if there are multiple downstream nodes, the flow from the upstream node is distributed evenly among them.

A merchant on a specific node performs the following two things: (1) increasing the $TPR$ of the country at that node, and (2) determining the direction of the flow to the home node. First, in Figure 2 (a), the $TPR$ is 10% for these nodes, and thus, the flow from each node are $(1 + 0.15 + 1.5) * 10\% = 0.265$ from node 1, and $(1+2.0)*10\% = 0.3$ from node 2. As we assumed, the value from node 2 to the home node is $0.3/2 = 0.15$ due to its multiple downstream nodes. Next, in Figure 2 (b), the merchant on node 1 increases the $TPR$ of the country to double. As a result, the flow from node 1 to home node increases from 0.265 to $0.265 * 2 = 0.53$. Finally, in Figure 2 (c), the merchant on node 2 increases the $TPR$ of the country and ensures that all outgoing values move toward the home node. Here, the value moves toward the home node increases from 0.15 to $(2.0 + 1) * 20\% = 0.45$, and toward the node 1 decreases to 0. Consequently, the value from node 1 decreases from 0.265 to $(1 + 1.5) * 10\% = 0.25$. Thus, in this example, the $d_{best}$ is to locate merchant on the node 2.

| node | local value | TPR without M | TPR with M |
|---|---|---|---|
| 1 | 1.5 | 10% | 20% |
| 2 | 2.0 | 10% | 15% |

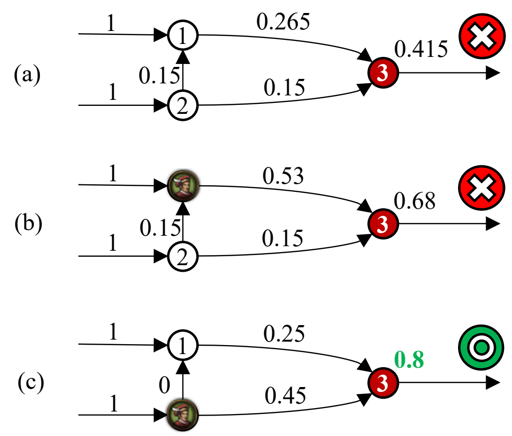



Figure 2: Example of the locating scenario. The red circle represents the home node of the country mentioned in the question. Each arrow means a upstream-downstream relationship. M means a merchant.

**Building scenario:** In the building scenario, an LM analyzes the supply chain and determines what building should be expanded in order to reduce the price of specific goods. The supply chain is composed of two different components:

- A set of buildings of which that consumes some goods to produce some other goods.
- A set of goods, each of whose price is decided

[2]Calculation of $TPR$ is on Appendix A.

by its supply and demand.

To reduce the price of specific goods, it is necessary to enlarge their supply.

Figure 3 (a) shows an example supply chain for furniture with two buildings that have different production methods. Each building cannot receive more goods than the maximum input, which is listed on the table in 3 The output for a building is calculated as (sum of max input / sum of max output) * (sum of input). For example, in Figure 3, building 1 and building 2 produces 40 and 45 pieces of furniture, respectively.

Expanding a building will increase both its max input and max output values, which means it can receive more input and generate a greater output. Figure 3 (b) illustrates the amount of output when both building 1 and building 2 are expanded to double. In the case of building 1, the supply rises to 80. However, for building 2, the supply only grows to 73. This discrepancy arises because the woods used in building 1 are supplied at 40, whereas the hardwoods used in building 2 are in short supply, at just 25. Thus, in this example, the $d_{best}$ is to enlarge building 2.

| building | max input per building | | max output per building |
|---|---|---|---|
| | wood | hardwood | furniture |
| 1 | 40 | - | 40 |
| 2 | 20 | 20 | 45 |

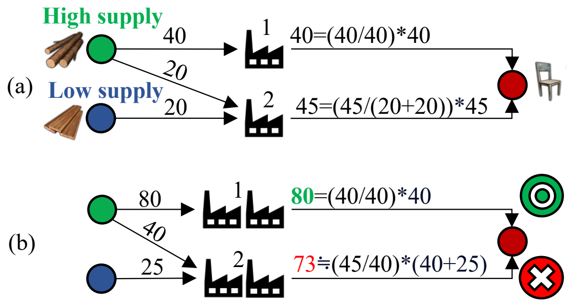



Figure 3: Example of the building scenario. Each circle represents goods, and the factory image represents a building. The red circle indicates goods needing a price reduction, furniture in this example. The yellow arrow represents the quantity of goods produced in the building. The green and blue circles represent wood, and hardwood respectively.

### 4.2 Data Collection

To collect game data, we select the earliest starting point provided by each of the games as a save-file and preprocess them by a game data parser to extract data. In order to control the quality of questions, we consider the following points: (1) For the locating scenario, we create one problem for each country. As previously explained, profits are determined by the trading power of each country. Hence, countries with low trading power might have minimal impact on decisions and are not chosen for question formulation. (2) For the building scenario, we formulate problems where there are decisions to expand existing buildings that can compensate for the insufficient supply of goods.

### 4.3 Simulator

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. First, in the actual game setting, various random events occur that can sway the results. It is hard to be sure that a decision validated in the game is always the best decision for our problem because of the randomness of the actual game. Secondly, since Europa Universalis IV and Victoria 3 are not open-sourced, it is impossible to open them as benchmark validation programs. Therefore, we develop simulators for each scenario on DQA, which can validate the results of decisions deterministically, and we utilize them as annotators for our benchmark.

### 4.4 Dataset Statistics

Finally, DQA consists of a total of 140 question and data pair: with 81 for the locating scenario and 59 for the building scenario. Each data in DQA is provided by the Cypher Query Language (CQL) (Francis et al., 2018) file. Table 1 shows the basic statistics of the data in each scenario.

Table 1: Basic statistics for each scenarios on DQA. $V$ and $E$ mean vertices and edges respectively.

| Statistics | Locating | Building |
|---|---|---|
| # of $<Q,D>$ pairs | 81 | 59 |
| Avg. # of $V$ per pair | 745 | 240.95 |
| Avg. # of $E$ per pair | $1,639$ | 504.72 |

## 5 Methodology: PlanRAG

### 5.1 Planning for Decision QA

As we explained in Section 3, a data analysis for Decision QA is composed of multiple small data analysis steps. To conduct Decision QA through one-step retrieval, an LM should combine these small data analysis tasks, each of which performs a separate role, into a single data analysis process. It is challenging for an LM. For the example in Figure 1, an LM should generate a complex query, as shown in Appendix B for one-step retrieval. In the iterative RAG technique, which could address this issue, an LM determines what data is required in each retrieval iteration. In terms of Decision QA, this retrieval can be translated as reasoning what data analysis is needed in each retrieval iteration, which is challenging because each reasoning requires understanding previous data analyses and the problem simultaneously. This approach is useful for the situation where each retrieval depends on the previous retrieval, such as multi-hop QA. However, in Decision QA, an LM determines a data analysis by examining the data schema, so it is possible to predict which data will be retrieved. Hence, there is no need to conduct reasoning for data analysis in every retrieval.

In this paper, we define a *plan* as the data analysis required for every iterative retrieval, and *planning* as the process that generates a plan. With a single planning, an LM can generate the plan for all iterative retrievals. This reduces the reasoning cost and leads to more accurate data analysis.

Figure 4 (a) represents the steps to solve Decision QA using the iterative RAG. In the first retrieval of this case, the LM obtains the upstream nodes of Deccan. In the second retrieval, the LM obtains the trading nodes having trading power for the country "BAH". For the third retrieval, the LM should analyze the profit of Doab, following the prior processes. However, it conducts the analysis that conflicts with previous retrievals, leading to incorrect results.

In contrast, if a planning is conducted before retrieving and each retrieval is done with corresponding plan, the retriever can generate a query that satisfies the necessary analysis. Figure 4 (b) illustrates the steps to solve Decision QA using the retrieval-augmentation method that includes planning. Unlike Figure 4 (a), since each retrieval follows the plan generated in the planning, it can consistently conduct data analysis.

### 5.2 PlanRAG: Plan-then-Retrieval Augmented Generation

In PlanRAG, the role of the generative language model is expanded to include planning. It is composed of the following three processes: (1) planning for data analysis, (2) retrieving for access external data, and (3) answer generating.

**Planning:** This process is an essential part of our approach and significantly distinguishes our technique from the existing retrieval-augmenting techniques (Lewis et al., 2020; Jiang et al., 2023b; Trivedi et al., 2023), that are primarily composed of retrieving and generating only. In the planning process, an LM generates an initial plan for data analysis by understanding the question and data schema. The initial plan generated from the planning process contains the order for data analysis retrieval. Figure 4 (b) provides an example of the initial plan in our technique.

Since the initial plan is not based on retrieved data, it may not remain valid until the answer generating process is done. To address this issue, an LM examines whether the existing plan remains valid after retrieving data, and performs replanning if it is no longer valid.

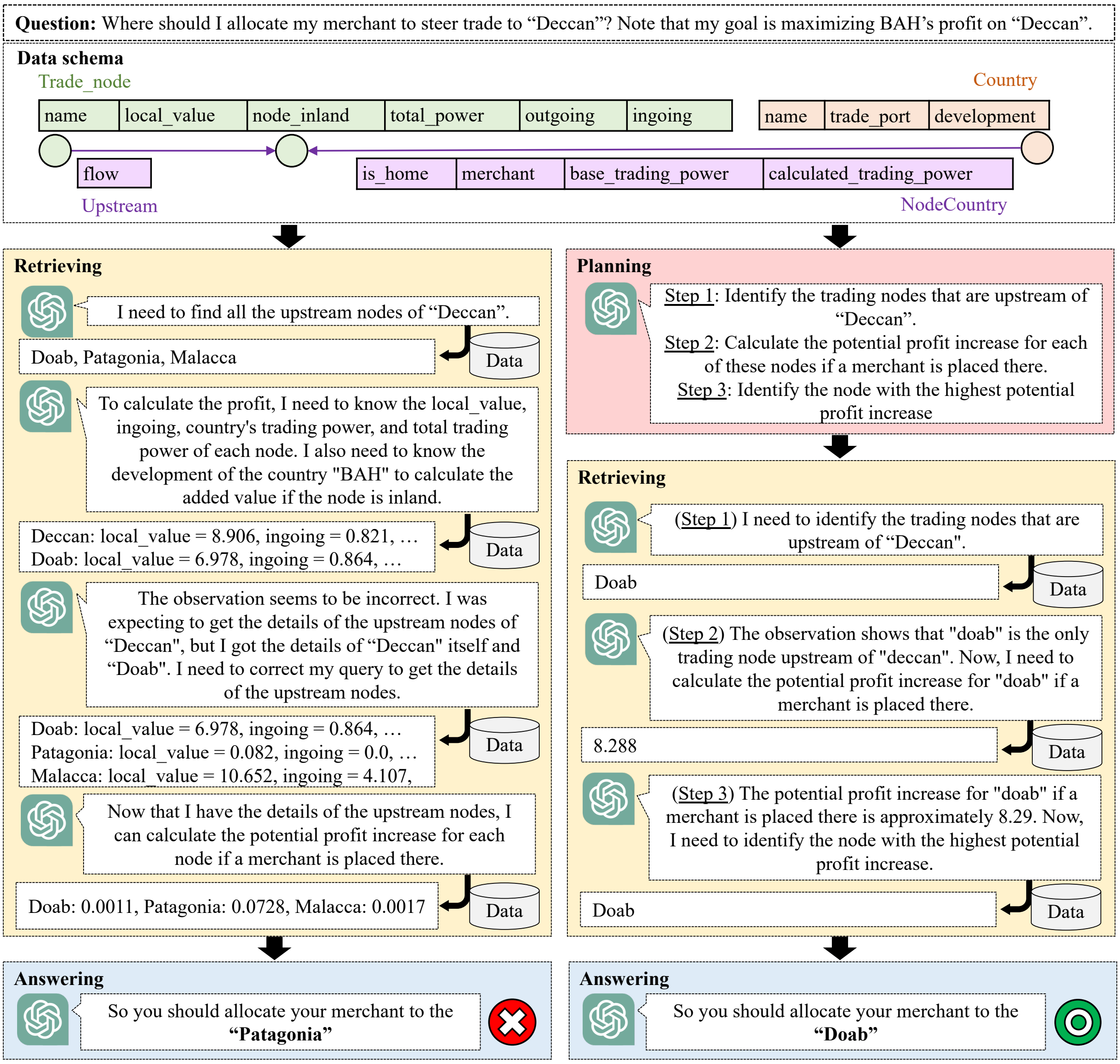
**Retrieving:** In this process, an LM generates the necessary data query and pose to retrieve, similar to existing iterative RAG studies(Yao et al., 2023; Jiang et al., 2023a). However, in previous studies, an LM determines which data analysis should be performed based on question and data schema in every retrieval iteration. In contrast, in PlanRAG, an LM simply generates a data analysis query without reasoning about which data analysis should be performed but rather following a previously generated plan. To accomplish this, an LM receives the generated plan explicitly. Figure 4 (b) explains how an LM performs data retrieving by the generated plan.

**Answer generating:** In this process, an LM generates an answer by understanding retrieved data, also similar to existing iterative RAG studies. Before the answer generating process, in previous iterative RAG methods, an LM a performed specific number of retrieval (Trivedi et al., 2023), or it determined whether the answer generating process should be executed in each retrieval (Yao et al., 2023). In contrast, in PlanRAG, an LM initiates the answer generating process if the plan has been executed completely. The answering process in Figure 4 (b) displays how an LM generates an answer just after the plan has been fully executed.

## 6 Experiments

### 6.1 Experimental Setup

To compare the single-turn RAG technique, the iterative RAG technique and PlanRAG technique,




**(a) Retrieval augmentation** **(b) Plan-then-Retrieval augmentation**

Figure 4: The process to solve Decision QA by (a) Retrieval augmentation technique, and (b) Plan-then-Retrieval augmentation technique. This example comes from the locating scenario on the DQA benchmark.

we implemented retrieval-augmented LMs following each technique and applied them to DQA in a single run. We utilized ReAct prompt (Yao et al., 2023) for implementing single-turn RAG based LM (SingleRAG-LM), iterative RAG-based LM (IterRAG-LM) and ReAct with planning step for implementing PlanRAG-based LM (PlanRAG-LM). These LMs are developed by LangChain[3] library and GPT-4(OpenAI, 2023) with a zero temperature. Prompts are provided in Appendix C. The answer provided by LM was considered correct if it was semantically identical to the answer on DQA. Otherwise, we considered it incorrect.

[3]https://langchain.readthedocs.io/en/latest

## 6.2 Results and Analysis

Our experimental results are described in Table 2. In our experiment, PlanRAG-LM demonstrated an accuracy of 55.6% in the locating scenario and 54.2% in the building scenario. These are, respectively, 29.7% and 25.4% higher than SingleRAG-LM, and 12.4% and 1.8% higher in accuracy than IterRAG-LM. Furthermore, the results for PlanRAG-LM without replanning show a decrease of 5.0% and 6.7% compared to PlanRAG-LM in the two scenarios. These results indicate that our technique, PlanRAG, is more suitable for solving Decision QA compared to iterative RAG methods, and we have confirmed that replanning is beneficial

Table 2: Performance comparison on the locating scenario and building scenario. RP means replanning.

| Techniques | Locating | Building |
|---|---|---|
| **Single-turn RAG** | | |
| SingleRAG-LM | 25.9 | 30.5 |
| **Iterative RAG** | | |
| IterRAG-LM | 43.2 | 54.2 |
| **PlanRAG (ours)** | | |
| PlanRAG-LM | **55.6** | **55.9** |
| PlanRAG-LM w/o RP | 50.6 | 49.2 |

for PlanRAG. To gain insights into the effectiveness of planning, we conducted a more in-depth analysis of the results of IterRAG-LM and PlanRAG-LM in both the locating and building scenarios.

Figure 5 shows the accuracy of IterRAG-LM and PlanRAG-LM in each scenario, depending on how many retrieving steps IterRAG-LM performed before Answering. We interpret these results by following three parts: (1) Single Retrieval (SR) problems in the locating scenario, (2) SR problems in the building scenario, and (3) Multiple Retrieval (MR) problems.

First, in the SR problems of the locating scenario, there was a significant increase in accuracy from IterRAG-LM to PlanRAG-LM. This improvement can be attributed to the characteristics of SR problems within the locating scenario. In this scenario, SR problems constitute a small portion and exhibit lower accuracy compared to the overall accuracy of IterRAG-LM. This indicates that the problems in the locating scenario are difficult to address with a single retrieval. PlanRAG-LM, on the other hand, can recognize the need for multiple retrievals in these problems. Leading to higher accuracy compared to IterRAG-LM.

In contrast, SR problems of the building scenario constituted a significant portion and exhibited high accuracy. It indicates that the building scenario contains a significant portion of problems that could be solved with a single retrieval. Since planning could be an unnecessary process in these problems, PlanRAG shows low accuracy on SR problems compared to IterRAG-LM.

Lastly, in the case of MR problems, regardless of the scenario, the accuracy increased when performing PlanRAG rather than the iterative RAG. This aligns with the discussion in Section 5.1, which suggests that planning is advantageous when conducting multiple retrievals. Through these analyses,

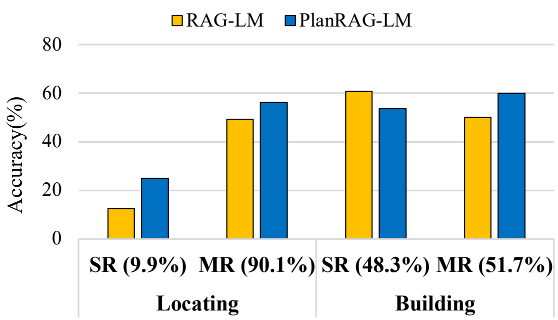



Figure 5: The accuracy of IterRAG-LM and PlanRAG-LM in each scenario is based on the number of retrieval iterations in IterRAG-LM. SR (Single Retrieval) refers to the case where IterRAG-LM performs one data retrieval and then answers, while MR (Multiple Retrieval) refers to the case where it answers after performing multiple data retrievals. The values inside the parentheses for SR and MR represent the proportion of the total questions that correspond to SR and MR, respectively.

we have confirmed that PlanRAG is more robust in Decision QA requiring complex data analysis processes multiple times.

## 7 Conclusions

In this paper, we explored the capability of LLM as a solution for decision making. Firstly, we introduced a new decision making task, **Decision QA**, which requires data analysis to make the best decision, and provided its benchmark, referred to as **DQA**. The DQA benchmark, designed to evaluate Decision QA performance, was constructed by accumulating data from two video games. Furthermore, we pointed out that the existing iterative RAG methods are not suitable for solving Decision QA, and suggested a plan-then-retrieval augmented generation technique, **PlanRAG**. To validate the effectiveness of our PlanRAG on Decision QA, we adopted both the iterative RAG-based LM and the PlanRAG-based LM to DQA. Through experiments, we confirmed that the PlanRAG-based LM exhibited superior performance in Decision QA that requires iterative retrieval, compared to the iterative RAG-based LM. Through deep analysis, we concluded that PlanRAG is more robust in Decision QA scenarios that require complex data analysis.

## 8 Limitations

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

First, in this study, we focused solely on Decision QA that uses graph-structured data. Decision QA based on other data formats, such as tabular or hybrid data, could be explored in future research.

Next, in this paper, we proposed techniques from a high-level RAG technique perspective that should be considered when solving Decision QA. Therefore, we did not address the low-level methods necessary 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, but it is not covered in this paper. These areas should also be addressed in future works.

## 9 Ethical Considerations

Language models have a hallucination issue and can potentially generate biased answers. Retrieval-augmented methods we have discussed in our study, 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)[4] 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"[5], we employed the term "BAH," and instead of "The Papel States"[6], we used "PAP" as terminology.

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

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

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

## 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.

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

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.

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.

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.

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*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

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

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.

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.

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

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

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.

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.

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.

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland. Association for Computational Linguistics.

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 TPR Calculation for the Locating Scenario

(a)

| country | development | home node |
|---|---|---|
| C | 24 | 3 |

(b)

| node | local value | $TP_T$ | $TP_C$ | is inland |
|---|---|---|---|---|
| 1 | 1.5 | 100 | 10 | True |
| 2 | 2.0 | 40 | 4 | False |
| 3 | 3 | 100 | 10 | False |

(c)

| node | local value | TPR without M | TPR with M |
|---|---|---|---|
| 1 | 1.5 | 10% | 20% |
| 2 | 2.0 | 10% | 15% |

(d)

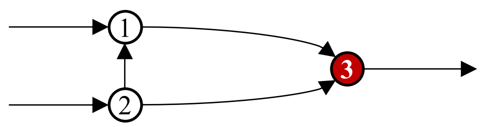



Figure 6: The trading network with the country table (a) and the node table (b). The $TPR$ value in the table (c), which is originally in Figure 2 could be generated by the table (a), the table (b) and the trading network (d). $TP_T$, $TP_C$, and M mean the total trading power and the trading power of the country, and a merchant respectively.

In this Section, we explain the calculation of $TPR$ values for the table (c) in Figure 6 using a portion of the table provided in the actual locating scenario, which are provided on the table (a) and the table (b) in Figure 6.

First, the situation without a merchant, the $TPR$ of the country on the node is calculated as $TPR = TP_C/TP_T$, by the definition which we mentioned on Section 4.1. For example, $TPR$ on node 1 is calculated as $10/100 = 10\%$

Next, the situation with a merchant, we should calculate the trading power increment of the country by merchant to calculate $TPR$. In our scenario, the trading power by a merchant ($TP_M$), provided as follows:

- If the located node or the home node is inland, then $TP_M = 2+\text{min(development/3, 50)}$
- Otherwise, $TP_M = 2$.

For example, in Figure A, $TP_M = 2+\min(50, 24/3) = 10$ if a merchant is located on the node 1. Therefore, $TPR$=(increased trading power of the country)/(total trading power)=$TP_M + TP_C/TP_T = 10 + 10/100 = 20\%$

## B Cypher Query for the Locating Scenario

```
MATCH
        (n:Trade_node)<-[:UPSTREAM]-(m:Trade_node),
        (c:Country {name: "BAH"})-[r:NodeCountry]->(m),
        (c)-[rr:NodeCountry{is_home: true}]->(n)
RETURN
        m.name,
        ((m.local_value+m.ingoing)*((r.calculated_trading_
power+(CASE WHEN m.node_inland THEN 2+CASE
WHEN c.development/3 > 50 THEN c.development/3 ELSE
50 END ELSE 2 END))/m.total_power) -
(m.local_value+m.ingoing)*(r.calculated_trading_power/m.t
otal_power))*100
        AS profit_diff_percent
```

Figure 7: Cypher query for the locating scenario. A language model can get potential profit by applying this query

## C Prompt setup

```
# 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, node_inland,
total_power, outgoing, ingoing}});
(m:Country {{name, trade_port, development}});
(Trade_node)-[r:UPSTREAM {{flow}}]-
>[Trade_node]
(Country)-[NodeCountry{{is_home,
merchant,base_trading_power,calculated_trading_po
wer}}]->(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: The prompt for the SingleRAG-LM retriever (based on ReAct, 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, node_inland, total_power, outgoing, ingoing}});
(m:Country {{name, trade_port, development}});
(Trade_node)-[r:UPSTREAM {{flow}}]->[Trade_node]
(Country)-[NodeCountry{{is_home, 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: The prompt for the SingleRAG-LM generator (based on ReAct, 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, node_inland, total_power, outgoing, ingoing}});
(m:Country {{name, trade_port, development}});
(Trade_node)-[r:UPSTREAM {{flow}}]->[Trade_node]
(Country)-[NodeCountry{{is_home, 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: The prompt for the IterRAG baseline (based on ReAct, Locating scenario).

# 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, node_inland, total_power, outgoing, ingoing}});
(m:Country {{name, trade_port, development}});
(Trade_node)-[r:UPSTREAM {{flow}}]->[Trade_node]
(Country)-[NodeCountry{{is_home, 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: The prompt for the PlanRAG baseline (based on PlanRAG, Locating scenario).