Ask, Clarify, Optimize: Human–LLM Collaboration for Smarter Inventory Control

Yaqi Duan* Yichun Hu† Jiashuo Jiang◇

Stern School of Business, New York University*
Johnson Graduate School of Management, Cornell University†
Department of Industrial Engineering and Decision Analytics,
Hong Kong University of Science and Technology◇

October 7, 2025

Abstract

Inventory management remains a challenge for many small and medium-sized businesses that lack the expertise to deploy advanced optimization methods. We present a framework that leverages large language models (LLMs) for decision support through natural-language interaction, combining a communicator agent that extracts structured parameters with an optimization agent that selects among classical (s, S) policies and deep reinforcement learning. To evaluate the system, we train a human imitator, a model fine-tuned on human–machine dialogues, that enables controlled yet realistic experiments. Results show that our agentic system reduces policy cost by 22.6% compared to GPT-4, demonstrating the promise of LLM-powered agentic systems for practical inventory management.

1 Introduction

Inventory management is a cornerstone of operations research, yet it remains difficult to implement effectively in practice. Decades of work have yielded theoretically elegant policies—from (s, S) thresholds to reinforcement learning formulations—but many small and medium-sized businesses still rely on informal judgment. Limited technical expertise, incomplete data, and the ambiguity of human decision-making often prevent these firms from deploying advanced optimization tools. As a result, inventory control in practice is frequently "good enough" rather than optimal, leaving significant efficiency gains untapped.

Recent advances in large language models (LLMs) bring a new opportunity. LLMs can interpret natural language, ask clarifying questions, and engage in dialogue, offering a natural interface between managers and decision-support systems. Yet, when used directly as solvers, LLMs have clear limitations: they may hallucinate, ignore domain-specific constraints, or rely on heuristics rather than provably effective algorithms. Purely LLM-based solutions risk producing plausible but suboptimal recommendations. What is needed is a hybrid approach that combines the flexibility of language models with the rigor of operations research.

Our approach builds on this premise. We develop an LLM-powered agentic system that (i) elicits and structures information from natural dialogue, (ii) leverages optimization algorithms such as (s, S) policies and deep reinforcement learning to compute policies, and (iii) adapts to user preferences while preserving interpretability and robustness. To evaluate this system under realistic yet reproducible conditions, we construct a *human imitator*: a language model fine-tuned on over a thousand human–machine dialogues, designed to mimic the ambiguity and bounded rationality of managers.

[1]Alphabetical order.

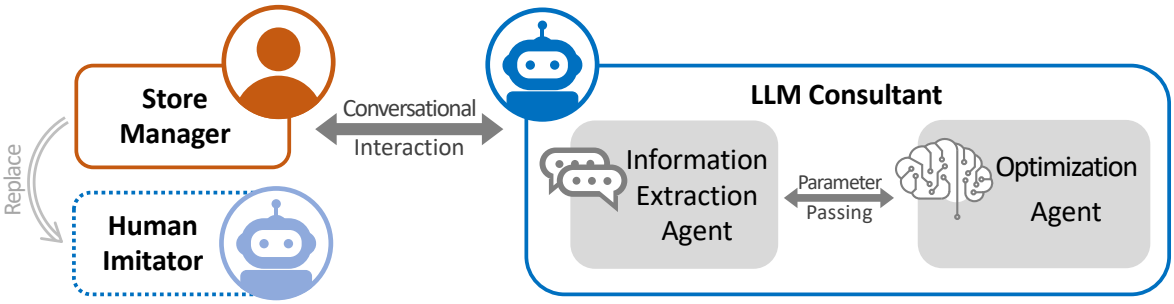

Figure 1: LLM-Agent Pipeline for Inventory Management

This design leads to three main contributions. We (1) introduce a human imitator model that enables scalable evaluation with realistic human-like input; (2) show how LLMs can be coupled with optimization routines to deliver adaptive, high-quality inventory policies; and (3) empirically demonstrate that both algorithmic grounding and information fidelity are essential for performance. Experiments highlight two central findings: grounding decisions in explicit optimization algorithms yields a 13.3% cost reduction compared to GPT-4 using the same structured inputs, and ensuring fidelity in parameter extraction reduces costs by a further 10.8% relative to GPT-4 operating through interactive dialogue.

Taken together, these results illustrate that LLMs can meaningfully augment operations not by replacing optimization theory, but by acting as intelligent interfaces that translate ambiguous human input into rigorous, algorithmically grounded decisions.

2 Framework for LLM-Powered Decision Support

Our framework integrates human-like problem input, structured information extraction, and adaptive optimization into a coherent pipeline (Figure 1).

2.1 Pipeline Architecture

The pipeline is composed of four interconnected components that together transform informal descriptions into optimized inventory policies.

Store Manager / Human Imitator. The entry point of the system is a natural-language description of the inventory problem. This role can be played by a real manager or by our trained human imitator, which reproduces the ambiguity, omissions, and bounded rationality characteristic of real-world decision-making.

LLM Consultant. The LLM consultant orchestrates the decision-support process and consists of two specialized agents:

Information Extraction Agent. This agent interprets the manager's informal input and incrementally converts it into structured parameters. Through conversational rounds, it asks clarifying questions, fills missing entries, and resolves conflicts until a complete and consistent parameter table is obtained.

Optimization Agent. Once the problem is fully specified, the optimization agent invokes appropriate solvers, such as the classical (s, S) policy or reinforcement learning algorithms, to generate inventory policies. The choice of solver depends on user-defined preferences, for example trading off expected cost against cost variance.

Together, these components establish an iterative, human-like workflow that mimics real managerial practice while remaining fully automatable.

2.2 Human Imitator: Modeling Manager Behavior

A central component of our system is the *human imitator*—a language model trained to reproduce the ambiguity and bounded rationality of retail managers. Real managers often misinterpret questions, provide rough estimates, or respond subjectively. Such inconsistency is not mere noise but a defining feature of real-world decision-making, and accounting for it is essential for robust evaluation.

Since recruiting human participants at scale is costly, we develop the imitator as a scalable substitute. Trained on real human–machine dialogues, it mimics the informality, omissions, and variability of managerial input, enabling reproducible experiments under realistic conditions.

Data Collection: We collected training data through an online platform where participants interacted with a base model. The model asked clarifying questions—about demand, perishability, costs, or lead times—and participants responded in natural language. The resulting dataset consists of prompt–response pairs, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, with each entry capturing a conversational slice (e.g., *"What is the shelf life of your product?"* $\rightarrow$ *"they last like 2 days usually"*). In total, *66* volunteers produced $N = 1{,}184$ samples.

Supervised Fine-Tuning (SFT): The imitator is modeled as a policy π_θ mapping prompts to responses and trained by maximum likelihood:

$$\max_{\theta} \quad \sum\nolimits_{(x_i,y_i)\in\mathcal{D}} \log \pi_\theta(y_i \mid x_i).$$

This objective aligns the imitator's conditional distribution with the empirical distribution of human replies, teaching it to "speak" like a manager with all the ambiguity and informality of natural input.

SFT was performed on *Qwen2.5-7B* using a single NVIDIA L4 GPU (24GB). Training lasted 524 minutes, with loss decreasing from 1.9234 to 0.3473. (See Appendix B for experimental detail.) The resulting imitator is released as a merged checkpoint of *Qwen2.5-7B* with *LoRA* adapters, so it can be invoked like any Hugging Face–hosted model. The repository is private for review but will be made public upon publication.

2.3 Information Extraction: Turning Dialogue into Parameters

We first let an Information Extraction Agent (in our experiments, built on GPT-4) converse with the manager and extract useful information from the conversation. The goal of the Information Extraction Agent is to populate a parameter table, which is initially empty, by interpreting the manager's plain-language responses.

The conversational process unfolds in rounds. In each round, the user interaction agent analyzes the Manager Agent's latest response and performs the following actions:

1. **Update Table:** If new, credible information is identified, it fills the corresponding blank in the table and updates the status to "defined."
2. **Detect Conflicts:** If the manager provides information that contradicts a previously defined entry, the agent flags the conflict for resolution.
3. **Formulate Question:** If the table contains undefined variables or identified conflicts, the agent generates a targeted question to clarify ambiguities or gather missing data. The conversation continues to the next round.

The dialogue terminates when all parameters are marked as "defined" and no conflicts exist. An example of a successfully filled table is shown in table 2.

Upon successful termination, the Information Extraction Agent saves two files: `extracted_params.csv` (the extracted parameters from the conversation), and `conversation_log.txt`. Finally, it invokes a solver, which uses the `extracted_params.csv` file to calculate and propose an optimal inventory policy.

2.4 Optimization: From Parameters to Policies

We now describe our optimization agent that solves the inventory problem directly and return the desired results to the user. The agent we develop has the ability to intelligently choose different algorithms to solve the problem, based on the user preference.

For the inventory problem, there are existing algorithms that can solve the problem efficiently, for example, the well-known (s, S) policy or the (deep) reinforcement learning (RL) algorithm that is based on the Markov Decision Process (MDP) formulation of the inventory problem. However, for a particular problem instance, different algorithms may have different performances on various metrics. For example, compared to the deep RL policy, the (s, S) policy may cause a higher expected cost but enjoy a lower variance. And the user may prefer lower variance than lower expected cost, or vice versa. Our optimization agent allows the user to input such a preference to intelligently select the appropriate result obtained from different algorithms. The entire procedure is illustrated in Figure 2. To be specific, the agent first ask for an input parameter λ, which specifies to what extent the user prefers lower expected cost than lower variance. The parameter λ is restricted to the range of $[-10, 10]$. Then, the LLM agent selects the appropriate policy π that minimizes the objective function

$$\text{Cost}(\pi) + \exp(\lambda) \cdot \text{Std}(\pi) \tag{1}$$

where $\text{Cost}(\pi)$ represents the expected cost under the policy π and $\text{Std}(\pi)$ represents the standard deviation of the cost under the policy π. Both $\text{Cost}(\pi)$ and $\text{Std}(\pi)$ can be approximated accurately with a finite number of samples. The agent then returns the obtained policy back to the user.

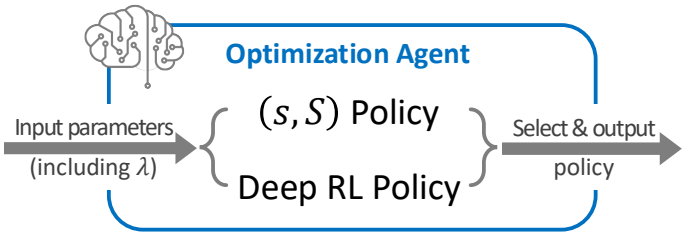

Figure 2: Optimization agent illustration

In our demonstration, we let our LLM agent to choose between the (s, S) policy and the deep Q-network (DQN) algorithm. The specific configurations for our policies are given in appendix D.

3 Evaluation

We now evaluate the performance of our proposed agentic system against baseline LLM benchmarks.

The evaluation process begins with a **Problem Generator Agent**, which procedurally generates unique inventory management problems. Each problem includes a high-level scenario (e.g., "Managing imported ingredient inventory for a premium restaurant business") and a set of ground-truth parameters that define the operational environment (see appendix E for detail). These parameters represent details a manager might intuitively grasp but cannot formally articulate. For each problem

instance, we compare the performance of policies derived from three distinct methods. Performance is measured by the average cost, calculated using the ground-truth parameters.

Smart Agentic System (Interactive): The **Human Imitator** uses ground-truth parameters as its knowledge base. It then simulates a manager who possesses domain knowledge but lacks technical expertise, conveying this information conversationally to our proposed agentic system to obtain an `extracted_params.csv` file and devise an inventory policy.

Vanilla GPT-4 (Interactive): The setup is identical to the first method, but the Human Imitator interacts with a vanilla GPT-4 model instead of our system to generate an inventory policy.

Vanilla GPT-4 (Direct Input): The `extracted_params.csv` file generated by our agentic system is provided directly to a vanilla GPT-4 model, which is then prompted to generate an inventory policy without any interactive dialogue.

We conducted an initial experiment using the instance with ground-truth parameters specified in table 2. The results show our system successfully extracted all critical parameters, while the interactive GPT-4 missed key details like demand distribution and setup costs. The Smart Agentic System (Interactive) policy also achieved the lowest average cost ($6146.95), compared to the Vanilla GPT-4 (Interactive) ($7942.51) and Vanilla GPT-4 (Direct Input) ($7086.69) policies.

Overall, our system achieved a 22.6% total cost reduction over the interactive GPT-4 baseline. This gain stems from two advantages: (i) **Superior Algorithms**: Our system's use of frontier algorithms accounted for a 13.3% cost reduction compared to the direct-input GPT-4, which used the same high-quality data but relied on general heuristics; and (ii) **Accurate Information Extraction**: Providing the direct-input GPT-4 with complete parameters yielded a 10.8% lower cost than the interactive GPT-4, highlighting the value of high-fidelity information gathering.

References

[1] A. AhmadiTeshnizi, W. Gao, and M. Udell. Optimus: Optimization modeling using mip solvers and large language models. *arXiv preprint arXiv:2310.06116*, 2023.

[2] M. Alvo, D. Russo, and Y. Kanoria. Neural inventory control in networks via hindsight differentiable policy optimization. *arXiv preprint arXiv:2306.11246*, 2023.

[3] H. Dehaybe, D. Catanzaro, and P. Chevalier. Deep reinforcement learning for inventory optimization with non-stationary uncertain demand. *European Journal of Operational Research*, 314(2):433–445, 2024.

[4] J. Gijsbrechts, R. N. Boute, J. A. Van Mieghem, and D. J. Zhang. Can deep reinforcement learning improve inventory management? performance on lost sales, dual-sourcing, and multi-echelon problems. *Manufacturing & Service Operations Management*, 24(3):1349–1368, 2022.

[5] I. Kaynov, M. van Knippenberg, V. Menkovski, A. van Breemen, and W. van Jaarsveld. Deep reinforcement learning for one-warehouse multi-retailer inventory management. *International Journal of Production Economics*, 267:109088, 2024.

[6] Y. Li, H. Zhao, H. Jiang, Y. Pan, Z. Liu, Z. Wu, P. Shu, J. Tian, T. Yang, S. Xu, et al. Large language models for manufacturing. *arXiv preprint arXiv:2410.21418*, 2024.

[7] K. Liang, Y. Lu, J. Mao, S. Sun, C. Zeng, X. Jin, H. Qin, R. Zhu, C.-P. Teo, et al. Llm for large-scale optimization model auto-formulation: A lightweight few-shot learning approach. *Congcong and Jin, Xiao and Qin, Hanzhang and Zhu, Ruihao and Teo, Chung-Piaw, LLM for Large-Scale Optimization Model Auto-Formulation: A Lightweight Few-Shot Learning Approach (June 28, 2025)*, 2025.

[8] Z. Liu, X. Li, S. Chen, G. Li, J. Jiang, and J. Zhang. Reinforcement learning with intrinsically motivated feedback graph for lost-sales inventory control. *arXiv preprint arXiv:2406.18351*, 2024.

[9] T. Lu, E. Garcia, and J. Lee. Optimizing supply chain demand forecasting and inventory management using large language models. 2024.

[10] H. Meisheri, V. Baniwal, N. N. Sultana, H. Khadilkar, and B. Ravindran. Using reinforcement learning for a large variable-dimensional inventory management problem. In *Adaptive learning agents workshop at AAMAS*, pages 1–9, 2020.

[11] H. Meisheri, N. N. Sultana, M. Baranwal, V. Baniwal, S. Nath, S. Verma, B. Ravindran, and H. Khadilkar. Scalable multi-product inventory control with lead time constraints using reinforcement learning. *Neural Computing and Applications*, 34(3):1735–1757, 2022.

[12] Z. A. Nazi and W. Peng. Large language models in healthcare and medical domain: A review. In *Informatics*, volume 11, page 57. MDPI, 2024.

[13] A. Oroojlooyjadid, M. Nazari, L. V. Snyder, and M. Takáč. A deep q-network for the beer game: Deep reinforcement learning for inventory optimization. *Manufacturing & Service Operations Management*, 24(1):285–304, 2022.

[14] H. Park, D. G. Choi, and D. Min. Adaptive inventory replenishment using structured reinforcement learning by exploiting a policy structure. *International Journal of Production Economics*, 266:109029, 2023.

[15] Y. Qi, J. Yin, J. Zhang, D. Geng, Z. Chen, H. Hu, W. Qi, and Z.-J. M. Shen. Leveraging llm-based agents for intelligent supply chain planning. *arXiv preprint arXiv:2509.03811*, 2025.

[16] Y. Quan and Z. Liu. Invagent: A large language model based multi-agent system for inventory management in supply chains. *arXiv preprint arXiv:2407.11384*, 2024.

[17] M. Selukar, P. Jain, and T. Kumar. Inventory control of multiple perishable goods using deep reinforcement learning for sustainable environment. *Sustainable Energy Technologies and Assessments*, 52:102038, 2022.

[18] N. H. Shah, D. Entwistle, and M. A. Pfeffer. Creation and adoption of large language models in medicine. *Jama*, 330(9):866–869, 2023.

[19] F. Stranieri, E. Fadda, and F. Stella. Combining deep reinforcement learning and multi-stage stochastic programming to address the supply chain inventory management problem. *International Journal of Production Economics*, 268:109099, 2024.

[20] F. Stranieri and F. Stella. Comparing deep reinforcement learning algorithms in two-echelon supply chains, 2023.

[21] N. N. Sultana, H. Meisheri, V. Baniwal, S. Nath, B. Ravindran, and H. Khadilkar. Reinforcement learning for multi-product multi-node inventory management in supply chains. *arXiv preprint arXiv:2006.04037*, 2020.

[22] L. Van Hezewijk, N. Dellaert, T. Van Woensel, and N. Gademann. Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem. *International Journal of Production Research*, 61(6):1955–1978, 2023.

[23] G. Wu, M. Á. de Carvalho Servia, and M. Mowbray. Distributional reinforcement learning for inventory management in multi-echelon supply chains. *Digital Chemical Engineering*, 6:100073, 2023.

[24] C. Zhou, J. Yang, L. Xin, Y. Chen, Z. He, and D. Ge. Auto-formulating dynamic programming problems with large language models. *arXiv preprint arXiv:2507.11737*, 2025.

A Related Work

LLMs are increasingly being adopted as versatile decision-support tools across many critical sectors. Within business and operations management, LLMs are leveraged to optimize processes, improve efficiency, and drive innovation li2024large. Significant recent research interest has focused on their ability to automatically formulate optimization problems from natural language descriptions ahmaditeshnizi2023optimus,zhou2025auto,liang2025llm. In supply chain management specifically, LLMs are used for tasks such as demand forecasting and logistics optimization lu2024optimizing, and recent work has also focused on developing agentic frameworks for advanced applications quan2024invagent,qi2025leveraging. In the healthcare domain, LLMs show remarkable potential to augment clinical decision-making by rapidly synthesizing patient information, generating differential diagnoses, and suggesting treatment plans, positioning them as powerful co-pilots for clinicians shah2023creation,nazi2024large. Similarly, the financial sector uses LLMs for a wide range of applications, from market analysis to risk management, with specialized models like FinGPT and BloombergGPT enabling nuanced sentiment analysis, automated report generation, and enhanced algorithmic trading strategies.

Deep Reinforcement Learning (DRL), when combined with deep neural networks, effectively addresses the high-dimensional state and action spaces inherent in inventory control (IC), alleviating the curse of dimensionality. Early studies demonstrated the feasibility of RL in IC. For instance, [13] utilized the Deep Q-network (DQN) to solve the beer distribution game, a widely studied supply chain simulation. Similarly, [4] employed the A3C algorithm to achieve heuristic-level performance, while [20] benchmarked multiple DRL methods, including A3C, PPO, and vanilla policy gradient (VPG), for inventory problems. Recent advancements have extended DRL to a variety of inventory control scenarios, such as managing non-stationary uncertain demand dehaybe2024deep, park2023adaptive, optimizing multi-product systems sultana2020reinforcement, selukar2022inventory, and handling diverse product types meisheri2020using, meisheri2022scalable. DRL has also been applied to complex supply chain structures, including multi-echelon systems wu2023distributional, alvo2023neural, liu2024reinforcement, stranieri2024combining, one-warehouse multi-retailer networks kaynov2024deep, and the stochastic capacitated lot-sizing problem van2023using.

B Human lmitator

The supervised fine-tuning (SFT) was performed on *Qwen2.5-7B* with *LoRA* adapters (rank $r = 16$, scaling $\alpha = 32$, dropout 0.05) under a quantized 4-bit (*NF4*) configuration with *bfloat16* computation. Training used the *Hugging Face Trainer* with *paged AdamW* (32-bit) optimization ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$), an initial learning rate of 2×10^{-4}, cosine decay scheduling, and a 3% warm-up phase. We trained for 3 epochs over $1,184$ samples, amounting to 222 optimization steps.

To accommodate the 24 GB memory of a single NVIDIA L4 GPU, the per-device batch size was set to 1, and gradient accumulation was applied over 16 iterations before each parameter update, yielding an effective batch size of 16. This strategy allowed us to simulate larger-batch optimization while remaining within the hardware limits. The training ran for 524 minutes in total.

As shown in Figure 3, the training loss exhibited fluctuations but decreased overall from 1.9234 at initialization to 0.3473 by the end of training. To support seamless integration into downstream tasks, we release the human imitator as a merged checkpoint of *Qwen2.5-7B* fine-tuned with *LoRA* adapters. By consolidating the adapters into the base model, the imitator can be loaded and invoked in exactly the same manner as any Hugging Face–hosted pretrained model, thereby removing unnecessary engineering overhead. The repository remains private during the review process but will be made publicly accessible upon publication.

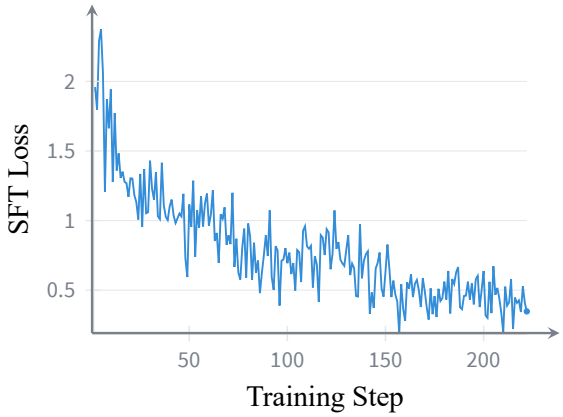

Figure 3: Training loss curve during supervised fine-tuning (SFT) of *Qwen2.5-7B*. The loss shows short-term fluctuations but an overall downward trend, decreasing from 1.9234 at initialization to 0.3473 by the end of training.

C Information Extraction

Table 1: Initial Parameter Table for the Information Extraction Agent

Parameter	**Value**	**Unit**	**Status**
time_horizon			undefined
demand_type		N/A	undefined
demand_distribution		N/A	undefined
perishability		N/A	undefined
state_transition_model		N/A	undefined
holding_cost			undefined
penalty_cost			undefined
setup_cost			undefined
lead_time			undefined
risk_tolerance		N/A	undefined

Table 2: Example of a Completed Parameter Table

Parameter	**Value**	**Unit**	**Status**
time_horizon	60	days	defined
demand_type	random	N/A	defined
demand_distribution	Poisson($\lambda = 10$) per day	N/A	defined
perishability	TRUE	N/A	defined
state_transition_model	lost_sale	N/A	defined
holding_cost	5	USD/unit/day	defined
penalty_cost	15	USD/unit	defined
setup_cost	100	USD/order	defined
lead_time	5	days	defined
max_inventory	100	N/A	defined
max_order	20	N/A	defined
risk_tolerance	3	1-10 scale	defined

D Optimization

(s, S) **policy.** The (s, S) policy is an inventory management approach where s is the reorder point and S is the order-up-to level. When the inventory level, which equals the summation of inventory position and all the orders in the pipeline, falls to or below s, a replenishment order is triggered to bring the stock back up to S. This policy balances ordering and holding costs by maintaining inventory between these thresholds, ensuring stock is replenished only when necessary. It is particularly effective in systems with positive lead times. While simple to implement, improper settings for s and S can lead to overstocking or stockouts. Our optimization agent simulates the expected cost and cost variance over different choices of (s, S) and selects the best (s, S) policy.

DQN policy. The DQN policy is based on the MDP formulation of the inventory problem, which is a reinforcement learning method that combines Q-learning with deep neural networks to enable agents to learn optimal policies in dynamic environments. Instead of using a Q-table, which is infeasible for large state-action spaces, DQN uses a neural network to approximate the Q-value function, which predicts the expected cumulative reward for each action given a

state. In our problem, the state refers to the composition of the inventory position, the current time period, and the orders in the pipeline. By iteratively updating the Q-network using the Bellman equation, DQN approximates the optimal policies using samples from the demand distribution. Note that DQN performs well even when the lead time is large, due to its ability to handle sequential decision making problems with high-dimensional input.

E Evaluation

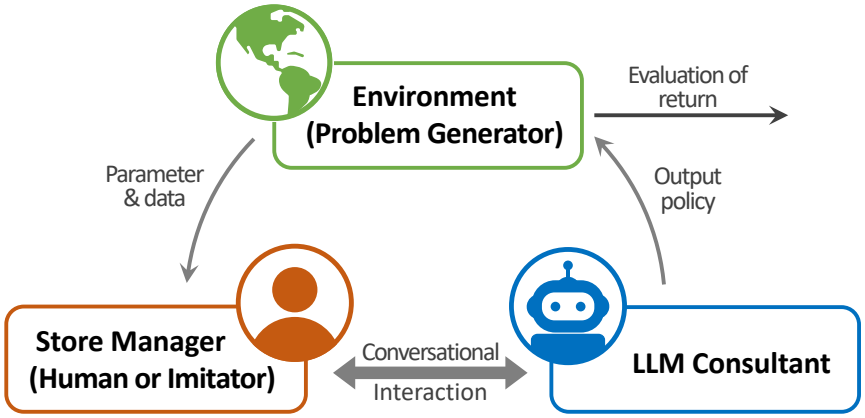

Figure 4: Framework for Evaluation

The parameters are as follows:

- `time_horizon`: The planning period for inventory decisions.
- `demand_type`: The nature of customer demand, categorized as either *deterministic* or *random*.
- `demand_distribution`: If deterministic, a constant value; if random, a statistical distribution (e.g., Normal($\mu = 30, \sigma = 10$)).
- `perishability`: A boolean value indicating if the items expire.
- `state_transition_model`: The outcome of a stockout, modeled as either a *lost sale* or a *backlog*.
- `holding_cost`: The cost to store one unit of inventory for a specific time period.
- `penalty_cost`: The cost incurred for each unit of unmet demand.
- `setup_cost`: The fixed cost associated with placing an order.
- `lead_time`: The time delay between placing an order and receiving it.
- `max_inventory`: The maximum allowable inventory level.
- `max_order`: The maximum quantity that can be ordered at one time.
- `risk_tolerance`: An integer on a 1–10 scale, where a lower value indicates higher risk aversion.