# LLMs Are Not Good Strategists, Yet Memory-Enhanced Agency Boosts Reasoning

**Yi Wu** *
Department of Computer Science
University of Chicago
yiwu@uchicago.edu

**Zhimin Hu** *
Department of Psychology
University of Wisconsin-Madison
hu436@wisc.edu

## Abstract

Strategic reasoning in dynamic environments, such as games, requires a balance between long-term strategy and short-term adaptations. Although specially trained agents can achieve superhuman performance, they often lack explainability and are highly dependent on extensive data for training. In contrast, approaches that leverage large language models (LLMs) benefit from few-shot learning but struggle to maintain strategic consistency. Drawing inspiration from existing cognitive models of human decision-making, which utilize various forms of memory, we introduce **EpicStar**, an LLM-based agent with cognitively inspired episodic and working memory modules. Episodic memory enables agents to draw on past experiences to formulate coherent long-term strategies, while working memory modulates active observation and decision variables essential for adaptation. We evaluated EpicStar in the strategy game StarCraft II, where it competes effectively against built-in agents at Level 6 difficulty, surpassing its predecessor at Level 5 with a smaller token budget. Our approach not only enhances adaptability, but also ensures strategic consistency, demonstrating the pivotal role that cognitive memory can play in strategic reasoning.

## 1 Introduction

Strategic reasoning in partially observable and dynamic environments is a formidable challenge, necessitating a delicate balance between long-term strategy and immediate rewards. In such contexts, agents engage in sequential decision-making tasks, navigating environments where they can only observe a limited portion of the full state.

Video games, particularly complex strategy games like StarCraft II, have become vital platforms for testing strategic reasoning due to their controllability and complexity. Despite large language models (LLMs) that demonstrate human-like behavior in various reasoning tasks Hu et al. (2024c); Mukherjee et al. (2024); Hu et al. (2024a), their application in environments like StarCraft II has faced challenges. Notable attempts, such as TextStarCraft II Ma et al. (2024), leverage long context windows to capture detailed state information and support complex reasoning loops. However, they still struggle with maintaining coherent plans and reusing previous experiences effectively. A key limitation of this approach, and similar ones Hu et al. (2024b); Shao et al. (2024); Qi et al. (2024), is the oversimplified assumption that decisions rely solely on the current state or are confined to immediate dependencies. This assumption fails to address the long-term dependencies that are fundamental to strategic gameplay and real-world decision-making Gershman & Daw (2017). Furthermore, as noted in Kambhampati et al. (2024), LLMs primarily function as format translators and are not inherently equipped for planning and reasoning tasks. Increasing the complexity of reasoning prompts may not necessarily improve performance.

In contrast, human decision-making integrates immediate environmental feedback with past experiences, learned knowledge, and future predictions Biderman et al. (2020); Philiastides et al. (2010).

*Equal contribution. Order is decided by the alphabet of the first name. Each person reserves the right to list their name first.

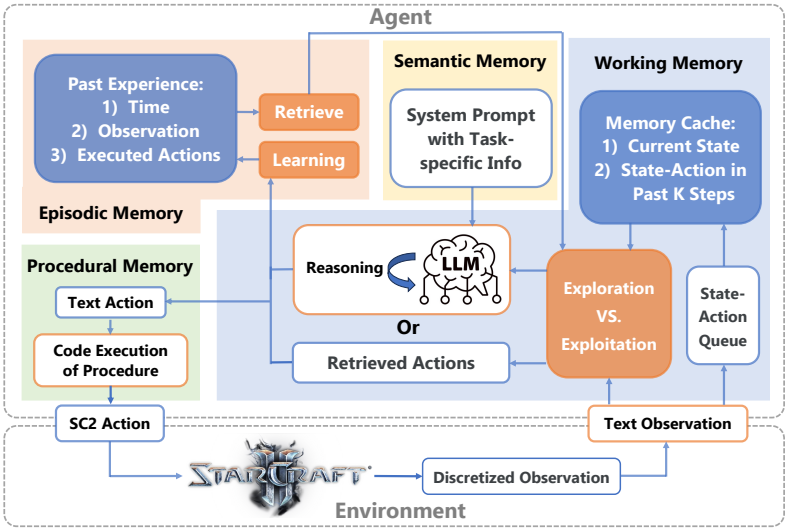


Figure 1: **Overview of the EpicStar architecture.** The agent consists of four memory components. Episodic memory stores gameplay episodes for learning and retrieval. Working memory balances coherent planning and dynamic adaptation by retrieving episodes and proposing exploratory actions through the LLM. Semantic memory provides task instructions and tailored game knowledge for LLM reasoning. Procedural memory maps textual actions to StarCraft II actions.

In particular, the theory of cognitive architecture, notably the Soar architecture Laird et al. (1987), suggests that intelligent agents are supported by complex interactions of multiple cognitive modules consisting of a decision procedure, working memory, and long-term memory that work together to enable flexible learning and decision-making.

Although scaling at inference time and engineering prompt tricks have become a powerful and widely adopted approach in the LLM community Snell et al. (2024), our research pivots towards a system-level agent design. Oriented by principles of cognitive architecture Sumers et al. (2023) and the critical role of memory in decision-making Shohamy & Daw (2015), we introduce **EpicStar**, a novel method that integrates episodic memory and working memory modules with LLM agents to improve strategic decision-making. Specifically, episodic memory stores past gameplay data, enabling agents to sustain coherent strategies by leveraging ”optimal” actions from previous experiences. Concurrently, working memory retains active reasoning variables for short-term adaptation, including recent observations from games, actions retrieved from episodic memory, and decision results derived from LLM reasoning. Aligning with the naming of Laird et al. (1987), we formulate the system prompt as semantic memory, representing essential ’knowledge about the world and itself’, and the code procedure that mappings the LLM-generated text action to the game action as procedural memory, representing ’explicit knowledge written in the agent’s code’ as defined by Sumers et al. (2023). As shown in Figure 1, working memory serves as the central console to connect different memory modules with predefined functions.

We adapt the game interface and the system prompt from Ma et al. (2024) to serve as procedural and semantic memory, respectively, making minor adjustments for fair comparisons. Our experiments, comparing EpicStar to the Chain of Summarization (CoS) method in Ma et al. (2024), demonstrate that our method outperforms the existing baseline with significantly higher win rates across difficulty levels (Table 1) while consuming fewer tokens (Appendix D) and exhibits greater adaptability under different test conditions, as shown in the ablation study 3.4.

In summary, our contributions are as follows: (1) a novel framework inspired by the cognitive mechanisms underlying human decision-making, designed to enhance strategic reasoning in LLM-based agents. (2) an effective approach in balancing long-term strategy and short-term adaptation through the integration of episodic and working memory. (3) experimental evidence that both long-term and short-term memories are crucial for strategic reasoning, and even a small amount of high-quality memory episodes can significantly impact outcomes.

## 2 REASONING WITH MEMORY

### 2.1 TASK FORMULATION

Strategic reasoning within StarCraft II can be conceptualized as a sequential decision-making process characterized by the tuple $(S, A, T, R, Z, O)$. $S$ represents the actual state space of the world, where $S = \{s_1, s_2, \ldots, s_N\}$ is the set of all possible states the system can be in. $A$ represents the action space, and $A = \{a_1, a_2, \ldots, a_M\}$ is the set of actions the agent can take. In environments that are partially observable, the agent has access to an observation space $O = \{o_1, o_2, \ldots, o_K\}$. $T$ acts as the state transition function $T(s' \mid s, a) = P(s' \mid s, a)$ and $Z$ is the observation function, $Z(o \mid s', a)$, represents the probability of receiving observation $o$ after the agent takes action $a$ and transitions to state $s'$. In this framework, the agent does not receive a direct reward $R(s, a)$ but rather observes the outcome of the game—either a win or a loss. Unlike traditional reinforcement learning, we do not directly estimate the $T$ and $Z$, but through LLM reasoning and memory mechanisms to maximize the expected accumulated reward.

### 2.2 EPISODIC MEMORY RETRIEVAL

Given a set of gameplay data, we define a series of episodes $\{(t_i, o_i, a_i) \mid i = 1, 2, \ldots, e\}$, where $t_i$ represents the time in the game. The episodic memory, denoted as $M^{EC}(t, o, a)$, consists of episodes corresponding to victory games.

For each moment $(t, o)$ in a new game, we identify similar past moments by finding indices of episodes $D_{index}$ that closely match the current scenario from $M^{EC}$. Initially, we search our memory to obtain a subset ${M^{EC}}_t$ proximate to the present $t$. Subsequently, we compare the current game's scenario with each moment in ${M^{EC}}_t$, focusing on significant changes in game elements and their values. Ultimately, we sort $M^{EC}t$ by its relevance to $(t, o)$ and get its relative indices, $Dindex$. For more details, please refer to Appendix C. Thus, the retrieval function is defined as:

$$Re^{EC}(t, o) = \begin{cases} \left\{\text{EmptyAction}\right\}, \text{if } |{M^{EC}}_t| = 0 \\ \left\{{M^{EC}}_t(t_i, o_i, a_i) \mid i \in D_{index}[:n]\right\}, \\ \text{otherwise} \end{cases} \tag{1}$$

### 2.3 WORKING MEMORY

For every time step $t$, the observation queue, $Q_o$, captures recent $k_{max}$ observations. Inspired by frame skipping Kalyanakrishnan et al. (2021); Braylan et al. (2015), the working memory recalls the

**Algorithm 1** Working Memory Exploration and Exploitation - $W(\cdot)$

**Require:** Exploration Cool Down Frame $d_e$, Queue Pop Cool Down Frame $d_q$, Action Size $n$, Current Game Time $t$, Last Time to Explore $l_e$, Last Time to Pop Queue $l_q$

$(e_1, \ldots, e_n) \leftarrow Re^{EC}(t, o_t)$ ▷ Retrieve $n$ episodes
$a_t \leftarrow ExtractAction((e_1, \ldots, e_n))$ ▷ Select the first action
**if** $t \geq d_e + l_e$ **then**
  $l_e \leftarrow t$
  $o'_t = o_t \times Q_o(t)$
  $(a_1, \ldots, a_n) \leftarrow LLM(S, (o'_t))$ ▷ Propose $n$ exploration actions
  $Q_a.push((a_1, \ldots, a_n))$
**end if**
**if** $a_t =$ EmptyAction **and** $Q_a.size() > 0$ **and** $t \geq l_q + d_q$ **then**
  $l_q = t$
  $a_t \leftarrow Q_a.pop()$ ▷ Add the exploration action
**end if**
$t \leftarrow t + 1$
**return** $a_t$

past observations by a frame interval of $L$:

$$Q_o(t) = \{(o_{t-k}) \mid k = L, 2L, \ldots, k_{\max}L\} \tag{2}$$

where $o_{t-k}$ represents observation at time $t-k$. Before reasoning, we map the observation $o_t$ to an augmented observation space defined as: $o'_t = o_t \times Q_o(t)$. We set $k_{max} = 4$ and $L = 24$.

The semantic memory denoted as $S$, serves as a system prompt, while the querying of the LLM is represented by the function $LLM(\cdot)$. We engage in the Exploration and Exploitation process denoted as $W(\cdot)$ and initiate it with an exploration action queue, $Q_a$, as outlined in Algorithm 1.

In $W(\cdot)$, $Q_o(t)$ provides short-term historical information that allows the agent to analyze the state's tendency. In contrast, $Q_a$ stores actions for future interpolation into action sequences retrieved from episodic memory, enabling adaptive planning.

### 2.4 Semantic Memory

Semantic memory acts as an agent's reservoir of knowledge about the world and itself Sumers et al. (2023). The prompting of 'CoS 'from Ma et al. (2024), has proven to be an effective form of semantic memory. It provides the agent with the basic information necessary to understand the game state, including the agent's race, game units, and action spaces. However, it fails to account for feasible actions given resource constraints and does not connect with episodic memory. To address these limitations, we implement two modifications to the prompt in CoS:

- Feasibility Check: We integrate additional instructions into the LLM prompt, encouraging the generated actions to be executable within the available resource constraints.
- Strategy Alignment: To enhance the alignment of exploration actions with episodic memory, we prompt the LLM to generate descriptive information about the game strategy used in episodic memory and concatenate this information with the original CoS prompt.

These adjustments enhance the agent's decision-making capabilities by ensuring that actions are not only feasible but also strategically coherent. The details of the prompt can be found in Prompt 1 in Appendix B. Given the semantic memory, which serves as the system prompt, we present our algorithm for the agent in Algorithm 2.

### 2.5 Procedural Memory

Given the action $a_t$ returned from Algorithm 2, procedural memory translates this action into specific procedures within StarCraft II. We adapt the game interface from Ma et al. (2024) to serve as procedural memory, leveraging the same action spaces for fair comparison. To enhance the generalizability of these procedures, we soften the constraint of the attack and defense procedures, allowing for more flexible strategies in both actions. This minor modification provides the agent with increased potential and variety in its micro-level decision-making, enabling more dynamic and context-sensitive behavior in the game.

**Algorithm 2** EpicStar Agent

**Require:** StarCraft II Game Environment $env$, Working Memory $W(\cdot)$

$env.initialize()$,
$Q_a \leftarrow \emptyset, l_e \leftarrow 0, t \leftarrow 0, l_q \leftarrow 0$
$o_t \leftarrow env.observation()$ ▷ Initial observation
**while** $env$ is not terminated **do**
    $a_t \leftarrow W(t, Q_a, o_t, Q_o(t), l_e, l_q)$
    $t \leftarrow t + 1$
    $o_t = env.step(a_t)$ ▷ Next frame's observation
**end while**
**return** $env.game\_result()$

| Difficulty | Agent Type | Win Rate | PBR | RUR | APU | TR |
|---|---|---|---|---|---|---|
| Level 5 | CoS(GPT-3.5-Turbo) | 0.550 (11/20) | 0.0781 | 7875 | 0.7608 | 0.4476 |
| | CoS(GPT-4-Turbo) | 0.600 (12/20) | 0.0337 | 8306 | 0.7194 | 0.3452 |
| | EpicStar | **0.675 (27/40)** | 0.1211 | 9864 | 0.8123 | 0.2536 |
| Level 6 | CoS(GPT-3.5-Turbo) | 0.0833 | - | - | - | - |
| | EpicStar | **0.300 (12/40)** | 0.1089 | 10931 | 0.7865 | 0.2304 |

Table 1: Overall comparison of EpicStar with Chain of Summarization (CoS), a strong baseline proposed by Ma et al. (2024), in the TextStarCraft II environment against Level 5 and Level 6 built-in agents. The results show significant improvements over the baseline at both Level 5 and Level 6 built-in agents.

### 2.6 Episodic Memory Learning

Episodic memory typically encompasses both learning and retrieval processes. To complement previous retrieval mechanisms, we accumulate episodic memory through an expert rule-based agent, drawing inspiration from imitation learning Zare et al. (2024). This approach empirically accelerates the accumulation of episodic memory. We gather game episodes as the agent competes against built-in agents at Level 6 and 7 across 20 rounds. We then isolate the winning rounds and select five rounds of episodes, ultimately yielding a total of $4592$ episodes. To ensure our evaluation results do not result from copy-pasting the expert agent in specific maps, we will test our agent on maps different from those used for memory collection.

## 3 Experiments

We conducted a series of experiments to evaluate the performance of EpicStar in StarCraft II, comparing it against built-in agent opponents across varying difficulty levels, attack styles, and game maps. We utilized the game interface in Ma et al. (2024) to establish our experimental environment, adhering to all standard settings, including the observation and action spaces. The only modification involved softening previous action procedures to allow more flexible action control, as mentioned in section 2.5

### 3.1 Experimental Setup

Our evaluation of EpicStar was carried out in two phases: (1) a comparative analysis against baseline methods and (2) an ablation study to assess the impact of individual components.

For all experiments, we used the `gpt-4o-mini-2024-07-18` model as the backbone to expedite the evaluation times. Although the `gpt-4-turbo` model, as used in Ma et al. (2024), has been shown to outperform the `gpt-4o-mini` model OpenAI (2024b), we opted for the latter to ensure a faster experimental turnaround. The highest difficulty level tested in Ma et al. (2024) was the Level 6 built-in agent. As a result, we primarily conducted our experiments at difficulty levels 5 and 6. To ensure consistency, we adopted the results from Ma et al. (2024) without reproducing their experiments, thus mitigating potential performance degradation. Additional details regarding the experimental setup and metrics can be found in Appendix A.

### 3.2 Evaluation Metrics

To evaluate EpicStar's performance in StarCraft II, we employed several metrics, with Win Rate as the primary measure. Additional metrics focused on efficiency in resource usage and technology development within the game. Detailed definitions of these metrics are provided in Appendix A.2. For this study, Win Rate serves as the main indicator of agent performance.

| Difficulty | Agent Type | Win Rate | PBR | RUR | APU | TR |
|---|---|---|---|---|---|---|
| Level 5 | w/o exploration | 0.600 (24/40) | 0.0598 | 11778 | 0.7628 | 0.2577 |
| | w/o strategy | 0.650 (26/40) | 0.1292 | 11339 | 0.8179 | 0.2619 |
| | EpicStar | **0.675 (27/40)** | 0.1211 | 9864 | 0.8123 | 0.2536 |
| Level 6 | w/o exploration | 0.175 (7/40) | 0.0522 | 10020 | 0.6828 | 0.2393 |
| | w/o strategy | 0.125 (5/40) | 0.0992 | 13744 | 0.7497 | 0.2161 |
| | EpicStar | **0.300 (12/40)** | 0.1089 | 10931 | 0.7865 | 0.2304 |

Table 2: Overall comparison of EpicStar and two ablation agents across the TextStarCraft II environment against Level 5 and Level 6 built-in agents. We found that ablating either Exploration in Working Memory or Strategy Alignment in Semantic Memory degrades the performance.

### 3.3 First Phase: Baseline Comparison

In the first phase of our experiments, we compared EpicStar against the original CoS baseline Ma et al. (2024). The agent was evaluated against built-in agents at difficulty levels 5 and 6, with 40 evaluation rounds conducted at each level. Specifically, we tested the agent against various built-in game strategies—`timing`, `rush`, `power`, `macro`, and `air`—as outlined in Appendix A.3. These tests were conducted on two newly introduced maps for the 2024 Season, `Abyssal Reef LE` and `Ever Dream LE`. Each strategy-map combination was tested four times, resulting in a total of 40 rounds.

### 3.4 Second Phase: Ablation Study

To evaluate the contribution of individual components within EpicStar, we conducted an ablation study, selectively removing key elements while keeping all other experimental conditions consistent with Phase 1. The first ablation involved the Exploration component in working memory, where we relied solely on retrieved episodes for action decision-making. This allowed us to isolate and assess the role of exploration via LLMs within the framework. In a subsequent ablation, we removed the Strategy Alignment from the system prompt to examine the influence of semantic memory on the agent's reasoning and planning abilities.

## 4 Results

### 4.1 Result Analysis Against Baseline

We present a comprehensive analysis of EpicStar's performance against the built-in agents at various difficulty levels in TextStarCraft II Ma et al. (2024), as shown in Table 1. At Level 5, EpicStar achieves a win rate of 67.5%, while at Level 6, it secures 30.0%. In contrast, the strongest baseline, CoS (GPT-4-Turbo), achieves a win rate of 60.0% at Level 5. CoS (GPT-4-Turbo), however, struggles considerably at Level 6, with a win rate of just 8.3% (GPT-3.5-Turbo). Our method achieves a significant win rate gain compared to the baseline.

Beyond the Win Rate, we observe additional metrics that validate EpicStar's superior performance. The Average Population Utilization (APU), which measures the efficiency of utilizing the population cap, reveals that EpicStar outperforms both baselines with a value of 0.8123, compared to 0.7608 for CoS (GPT-3.5-Turbo) and 0.7194 for CoS (GPT-4-Turbo). A higher APU indicates more effective macromanagement, which directly contributes to the agent's strength in facilitating units in the game.

Furthermore, we note occasional discrepancies between the Population Block Ratio (PBR), Resource Utilization Ratio (RUR), and the Win Rate. These discrepancies arise because, in matches where EpicStar wins against the built-in agents, our agent does not accept the agents' surrender. As a result, the game is extended until EpicStar reaches the population cap and can no longer spend its resources effectively, which may lead to lower PBR and RUR values despite a win.

| Difficulty | Strategy | Win Rate | PBR | RUR | APU | TR |
|---|---|---|---|---|---|---|
| **w/o exploration** | | | | | | |
| Level 5 | Air | 0.375 (3/8) | 0.0432 | 13553.09 | 0.7394 | 0.2292 |
| | Macro | 0.875 (7/8) | 0.1172 | 11727.99 | 0.8559 | 0.2976 |
| | Power | 0.875 (7/8) | 0.0439 | 15669.35 | 0.7645 | 0.2649 |
| | Rush | 0.375 (3/8) | 0.045 | 7742.42 | 0.7133 | 0.2351 |
| | Timing | 0.500 (4/8) | 0.0497 | 10196.04 | 0.7409 | 0.2619 |
| Level 6 | Air | 0.625 (5/8) | 0.0649 | 12895.88 | 0.7849 | 0.2738 |
| | Macro | 0.250 (2/8) | 0.0604 | 12408.85 | 0.7068 | 0.2976 |
| | Power | 0.000 (0/8) | 0.0535 | 15562.77 | 0.676 | 0.2738 |
| | Rush | 0.000 (0/8) | 0.0492 | 5398.43 | 0.6461 | 0.1548 |
| | Timing | 0.000 (0/8) | 0.0332 | 3832.86 | 0.6001 | 0.1964 |
| **w/o strategy** | | | | | | |
| Level 5 | Air | 0.625 (5/8) | 0.1103 | 11256.01 | 0.8364 | 0.253 |
| | Macro | 0.875 (7/8) | 0.091 | 11527.99 | 0.8416 | 0.3095 |
| | Power | 0.625 (5/8) | 0.1462 | 12328.87 | 0.8057 | 0.2708 |
| | Rush | 0.500 (4/8) | 0.1319 | 10903.58 | 0.7795 | 0.25 |
| | Timing | 0.625 (5/8) | 0.1665 | 10679.89 | 0.8264 | 0.2262 |
| Level 6 | Air | 0.375 (3/8) | 0.0741 | 18453.70 | 0.7515 | 0.2381 |
| | Macro | 0.250 (2/8) | 0.0996 | 19231.90 | 0.7382 | 0.2381 |
| | Power | 0.000 (0/8) | 0.1144 | 21609.83 | 0.6988 | 0.2708 |
| | Rush | 0.000 (0/8) | 0.1124 | 1891.27 | 0.8415 | 0.1339 |
| | Timing | 0.000 (0/8) | 0.0953 | 7533.12 | 0.7186 | 0.1994 |
| **EpicStar** | | | | | | |
| Level 5 | Air | 0.375 (3/8) | 0.0815 | 10439.92 | 0.7624 | 0.2768 |
| | Macro | 1.000 (8/8) | 0.1144 | 8908.00 | 0.8677 | 0.253 |
| | Power | 0.875 (7/8) | 0.1568 | 9201.55 | 0.8308 | 0.247 |
| | Rush | 0.625 (5/8) | 0.1032 | 10105.12 | 0.7929 | 0.2827 |
| | Timing | 0.500 (4/8) | 0.1497 | 10666.53 | 0.8076 | 0.2083 |
| Level 6 | Air | 0.625 (5/8) | 0.1256 | 14401.79 | 0.8086 | 0.2679 |
| | Macro | 0.250 (2/8) | 0.1024 | 14727.84 | 0.7539 | 0.2708 |
| | Power | 0.375 (3/8) | 0.0902 | 17272.48 | 0.7735 | 0.2589 |
| | Rush | 0.125 (1/8) | 0.1126 | 2713.02 | 0.8094 | 0.1429 |
| | Timing | 0.125 (1/8) | 0.1134 | 5541.10 | 0.787 | 0.2113 |

Table 3: Detailed results across build-in strategies and maps on EpicStar and two agent types: w/o exploration and w/o strategy. Win Rate: wins / total games.

### 4.2 Result Analysis on Ablation Study

To further investigate the impact of Exploration in Working Memory and Strategy Alignment in Semantic Memory, we conducted ablation studies on these two components, as detailed in Section 3.4.

The results demonstrate that ablating either of these components significantly degrades the agent's overall performance, as shown in Table 2. At Level 5, removing Exploration reduces the win rate from 67.5% (EpicStar) to 60.0%, while removing Strategy Alignment lowers it to 65.0%. This indicates that both components contribute to the agent's performance, with Exploration playing a slightly more critical role in this setting. At Level 6, the performance degradation is even more pronounced: the win rate drops sharply to 17.5% without Exploration and 12.5% without Strategy Alignment, compared to 30.0% with EpicStar. These findings highlight the increasing importance of strategy coherence at higher difficulty levels, where misaligned exploration can negatively impact performance. Overall, these results underscore the indispensable roles of both Exploration and Strategy Alignment for achieving high performance across various difficulty levels.

In addition to these overall performance trends, we examined detailed round results in Figure 3 to understand why EpicStar outperforms the ablated agents, particularly at Level 6. We observed that EpicStar performs well in strategies, notably the `Power`, `Rush`, and `Timing` strategies in Level 6, where the ablated agents fail entirely. For instance, while the ablated agents fail to win any games

with the `Power` strategy, EpicStar secures a 37.5% win rate, demonstrating its superior ability to utilize available resources efficiently. Similarly, EpicStar achieves a 12.5% win rate in the `Rush` strategy, compared to 0% for the ablated agents. In the `Timing` strategy, EpicStar achieves a modest but significant win rate of 12.5%, whereas both ablated versions fail. These results illustrate the synergy between Exploration for actions and Strategy Alignment with memory episodes, allowing EpicStar to handle a wider range of strategic scenarios, particularly against aggressive playstyles, thus making it more robust in complex, high-difficulty environments.

## 5 Discussion & Future Work

Our study aims to offer an alternative perspective on designing robust and flexible reasoning agents. Despite advances in prompt engineering and the rapid development of stronger foundational models, we argue that agent design is a systematic engineering process, not merely an accessory to a specific LLM. We have demonstrated that significant enhancements can be achievable by intricately designing the system using prompts as control flows and integrating short-term and long-term memory mechanisms, without the need to upgrade the model or increase token consumption. We hope this design philosophy benefits the engineering field and underscores the importance of memory mechanisms, especially those derived from cognitive processes, as complexity in reasoning and planning tasks increases. Note that the naming of the memory components is designed to align with Soar cognitive architectures, as outlined by Laird et al. (1987). Our objective is not to replicate the full complexity of human memory systems; rather, our method draws conceptual inspiration from the essential foundations necessary for strategic reasoning.

However, constrained by time, our study is not without its imperfections. Successful reasoning involves the ability to evolve (Wang et al., 2023a; Chuang et al., 2023; Kumarappan et al., 2024; Kumar et al., 2020). Our framework has the potential to evolve over time, as we have shown that agents that combine exploration and episode retrieval can achieve improved performance than retrieval-only agents. Through episodic updates and curriculum learning Bengio et al. (2009), we anticipate that the agent will demonstrate an evolving trajectory from the outset. This aspect will be a focus of future experiments. Additionally, while our study included a comprehensive set of experiments, there remains substantial scope for improvement by tailoring hyperparameters. Lastly, although our framework was solely tested in StarCraft II due to time constraints, the performance results provide decent confidence as StarCraft II condenses a wide range of strategic conditions. However, we acknowledged the need for broader application and further validation under varied conditions in future work.

## 6 Related Work

### 6.1 Interactive Environments

Interactive environments employed to benchmark artificial intelligence (AI) algorithms exhibit varying degrees of complexity, depending on game knowledge and strategic depth. Well-established environments such as ALFWorld Shridhar et al. (2021), ScienceWorld Wang et al. (2022a), and BabyAI Chevalier-Boisvert et al. (2019) feature predefined tasks with limited adaptability. In contrast, open-world environments like Crafter and Minecraft provide opportunities for exploration but lack adversarial complexity. StarCraft II, however, introduces significant challenges for LLMs and agents, due to its intensive control demands, vast action spaces, intricate state representations, and long-term game trajectories.

### 6.2 LLM-based Game Agents & StarCraft II AI

LLM-based game agents can be categorized into three main approaches: (1) Prompt-based methods, such as ReAct Yao et al. (2022) and Reflexion Shinn et al. (2024), which iteratively refine strategies; (2) Supervised fine-tuning approaches, exemplified by E2WM Xiang et al. (2023) and Synapse Zheng et al. (2024), which leverage high-quality trajectory data; and (3) Reinforcement learning (RL) methods, including GLAM Du et al. (2022) and TWOSOME Tan et al. (2024), that optimize policies using Proximal Policy Optimization (PPO) algorithms Schulman et al. (2017).

| Work | AlphaStar | SCC | HierNet-SC2 | AlphaStar Unplugged | ROA-Star | Chain of Summarization (CoS) | EpicStar |
|---|---|---|---|---|---|---|---|
| **Method** | SL+RL+Self-play | SL+RL+Self-play | Data-mining+RL | Offline RL | SL+RL+Self-play | Prompt+Rules | Prompt+Memory Module+Rules |
| **Compute resource** | 12000 CPU cores, 384 TPUs | Linear | 4 GPUs, 48 CPU cores | unknown | 2× 64 V100 | 1 GPU and CPU (home computer) | 1 CPU (home computer), API |
| **Required replay** | 971,000 | 4,638 | 608 | 20,000,000 (20M) | 120,938 | 0 | 20 |
| **Strongest opponent conquered** * | Serral (One of the best pro gamers) | Time (IEM 2023 Champion) | Level 10 built-in game agent | AlphaStar BC agent | Hero (GSL Champion) | Level 5 built-in game agent | Level 6 built-in game agent |
| **Interpretability** | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| **Generalizability on maps and races** | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |

Table 4: Comparison of various methods in StarCraft II AI research. (* If the opponent is a bot, we consider "conquered" as win rate $> 10\%$)

StarCraft AI research has evolved significantly from the original StarCraft to the more complex StarCraft II, fueled by the introduction of PySC2 DeepMind (2017) and extensive replay datasets. AlphaStar Vinyals et al. (2019) demonstrated the potential of RL by achieving Grandmaster-level performance. While numerous subsequent studies have optimized gameplay through improvements in input processing Liu et al. (2021), the adoption of offline RL Mathieu et al. (2023), and other techniques, there remains limited exploration Ma et al. (2024); Li et al. (2024) of LLM-based agents in this domain. A comparative overview of these methods in StarCraft II AI research is provided in Table 4.

### 6.3 Reasoning & Planning with LLM

Recent advancements in LLMs have significantly enhanced their capabilities in reasoning and planning. Techniques such as Chain-of-Thought (CoT) and Tree-of-Thought (ToT) prompting facilitate step-by-step problem-solving and structured exploration Wei et al. (2022); Yao et al. (2023). ReAct integrates reasoning with action to improve decision-making, while DEPS leverages visual feedback for enhanced planning Yao et al. (2022); Wang et al. (2023b). Models like OpenAI's O1 and DeepSeek's R1 further push the boundaries of internal deliberation and open-source accessibility, excelling in tasks such as mathematics, science, and coding OpenAI (2024a); DeepSeek-AI et al. (2025).

Despite these advancements, challenges persist, particularly in long-term planning and maintaining consistency across reasoning chains Kambhampati et al. (2024). Self-consistency decoding enhances reliability by selecting the most frequent reasoning path, while models like Toolformer improve problem-solving by integrating external tools Wang et al. (2022b); Schick et al. (2023). Nonetheless, ensuring robustness, interpretability, and real-world generalization remains an open research challenge. In this context, LLM-based agents with memory systems may help address these challenges more effectively than relying solely on LLMs.

## 7 Conclusion

In this paper, we present a multi-module framework designed to enhance strategic reasoning in LLM-based agents. By integrating episodic and working memory systems, we enable agents to maintain coherent strategic trajectories and adapt dynamically to evolving game scenarios. Empirical results demonstrate that our approach not only outperforms existing models like TextStarCraft II Ma et al. (2024) in terms of performance, but also highlights the critical role of a well-orchestrated memory system in replicating human-like strategic reasoning. Our system, which leverages small but high-quality episodes, shows that even limited data can be maximized for significant performance gains when paired with a robust cognitive framework. This research underscores the importance of cognitive-inspired designs in developing AI that can navigate and excel in complex strategic environments, bridging the gap between artificial intelligence and human cognitive mechanisms.

## References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Natalie Biderman, Akram Bakkour, and Daphna Shohamy. What are memories for? the hippocampus bridges past experience with future decisions. *Trends in Cognitive Sciences*, 24(7):542–556, 2020.

Alexander Braylan, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen. Frame skip is a powerful parameter for learning to play atari. In *AAAI Workshop: Learning for General Competency in Video Games*, 2015. URL https://api.semanticscholar.org/CorpusID:194604.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

Yun-Shiuan Chuang, Yi Wu, Dhruv Gupta, Rheeya Uppaal, Ananya Kumar, Luhang Sun, Makesh Narsimhan Sreedhar, Sijia Yang, Timothy T Rogers, and Junjie Hu. Evolving domain adaptation of pretrained language models for text classification. *arXiv preprint arXiv:2311.09661*, 2023.

Google DeepMind. Pysc2 - starcraft ii learning environment, 2017. URL https://github.com/google-deepmind/pysc2. Accessed: 2024-12-27.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948. Accessed: 2025-02-07.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Samuel J Gershman and Nathaniel D Daw. Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual review of psychology*, 68(1):101–128, 2017.

Jennifer Hu, Kyle Mahowald, Gary Lupyan, Anna Ivanova, and Roger Levy. Language models align with human judgments on key grammatical constructions. *Proceedings of the National Academy of Sciences*, 121(36):e2400917121, 2024a.

Sihao Hu, Tiansheng Huang, and Ling Liu. Pokellmon: A human-parity agent for pokemon battles with large language models. *arXiv preprint arXiv:2402.01118*, 2024b.

Zhimin Hu, Jeroen van Paridon, and Gary Lupyan. Failures and successes to learn a core conceptual distinction from the statistics of language. The Evolution of Language: Proceedings of the 15th International Conference, 2024c.

Shivaram Kalyanakrishnan, Siddharth Aravindan, Vishwajeet Bagdawat, Varun Bhatt, Harshith Goka, Archit Gupta, Kalpesh Krishna, and Vihari Piratla. An analysis of frame-skipping in reinforcement learning. *arXiv preprint arXiv:2102.03718*, 2021.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024.

Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5468–5479. PMLR, 13–18 Jul 2020.

Adarsh Kumarappan, Mo Tiwari, Peiyang Song, Robert Joseph George, Chaowei Xiao, and Anima Anandkumar. Leanagent: Lifelong learning for formal theorem proving. *arXiv preprint arXiv:2410.06209*, 2024.

John E Laird, Allen Newell, and Paul S Rosenbloom. Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1):1–64, 1987.

Zongyuan Li, Yanan Ni, Runnan Qi, Lumin Jiang, Chang Lu, Xiaojie Xu, Xiangbei Liu, Pengfei Li, Yunzheng Guo, Zhe Ma, et al. Llm-pysc2: Starcraft ii learning environment for large language models. *arXiv preprint arXiv:2411.05348*, 2024.

Ruo-Ze Liu, Wenhai Wang, Yanjie Shen, Zhiqi Li, Yang Yu, and Tong Lu. An introduction of mini-alphastar. *arXiv preprint arXiv:2104.06890*, 2021.

Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Runji Lin, Yuqiao Wu, Jun Wang, and Haifeng Zhang. Large language models play starcraft II:benchmarks and a chain of summarization approach. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Michaël Mathieu, Sherjil Ozair, Srivatsan Srinivasan, Caglar Gulcehre, Shangtong Zhang, Ray Jiang, Tom Le Paine, Richard Powell, Konrad Żołna, Julian Schrittwieser, et al. Alphastar unplugged: Large-scale offline reinforcement learning. *arXiv preprint arXiv:2308.03526*, 2023.

Kushin Mukherjee, Timothy T Rogers, and Karen B Schloss. Large language models estimate fine-grained human color-concept associations. *arXiv preprint arXiv:2406.17781*, 2024.

OpenAI. Openai o1 system card, 2024a. URL https://openai.com/index/openai-o1-system-card/. Accessed: 2025-02-07.

OpenAI. Simple-evals: Lightweight evaluation framework. https://github.com/openai/simple-evals, 2024b. Accessed: 2025-02-01.

Marios G Philiastides, Guido Biele, Niki Vavatzanidis, Philipp Kazzer, and Hauke R Heekeren. Temporal dynamics of prediction error processing during reward-based decision making. *Neuroimage*, 53(1):221–232, 2010.

Siyuan Qi, Shuo Chen, Yexin Li, Xiangyu Kong, Junqi Wang, Bangcheng Yang, Pring Wong, Yifan Zhong, Xiaoyuan Zhang, Zhaowei Zhang, et al. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents. *arXiv preprint arXiv:2401.10568*, 2024.

Timo Schick, Jane Dwivedi-Yu, Wissam Siblini, Ali Remmy, Omkar Jagtap, Joachim Bingel, Thang Vu, Sebastian Ruder, Oliver Korrel, Damien Teney, et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Xiao Shao, Weifu Jiang, Fei Zuo, and Mengqing Liu. Swarmbrain: Embodied agent for real-time strategy game starcraft ii via large language models. *arXiv preprint arXiv:2401.17749*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Daphna Shohamy and Nathaniel D Daw. Integrating memories to guide decisions. *Current Opinion in Behavioral Sciences*, 5:85–90, 2015.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hILVmJ4Uvu.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022a.

Wenlong Wang, Yilun Zhou, Shixiang Shane Gu, Julie Shah, Joshua Tenenbaum, Bill Freeman, and Jiajun Liu. Deps: A dataset for evaluating planning and reasoning in real-world visual environments. *arXiv preprint arXiv:2302.01356*, 2023b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Maarten Bosma, Ed H Chi, Quoc V Le, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. Language models meet world models: Embodied experiences enhance language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=SVBR6xBaMl.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Karthik Narasimhan Cao, Karthik Narasimhan, Xavier Amatriain, and Yuan Liu. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Maryam Zare, Parham M Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 2024.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Pc8AU1aF5e.

# A Experimental Setup and Metrics Description

## A.1 Experimental Setup

**Agent and Opponent Configuration:** To maintain a consistent and controlled testing environment, LLM agents are set to play as Protoss against built-in agent-controlled Zerg opponents. This arrangement enables a systematic evaluation of strategic performance across varying difficulty levels. The difficulty settings are listed below.

**Parameter Configuration:** The temperature parameter is set to 0.1 to prioritize strategic decision-making over random actions.

**Game Version:** All experiments were conducted using the latest Patch `5.0.14.93333` of StarCraft II.

| Level | BLZ Difficulty | Approximate League Equivalent |
|---|---|---|
| 1 | Very Easy | Bronze (Low) |
| 2 | Easy | Bronze (Mid) |
| 3 | Medium | Bronze (High) |
| 4 | Hard | Silver (Low) |
| 5 | Harder | Silver (Mid) |
| 6 | Very Hard | Silver (High) |
| 7 | Elite | Gold (Low) |
| 8 | Cheat Vision | Gold (Mid) |
| 9 | Cheat Money | Gold (High) |
| 10 | Cheat Insane | Platinum (Low) |

Table 5: StarCraft II Built-in Agent Levels and Approximate League Equivalents

### A.2 Evaluation Metrics

Our evaluation framework for TextStarCraft II Ma et al. (2024) builds upon StarCraft II's established player performance analytics, incorporating tailored modifications to comprehensively assess LLM agent gameplay strategies.

**Win Rate:** This is the primary performance indicator that evaluates the agent's performance in the game. It is calculated as the percentage of victories relative to the total games played.

**Population Block Ratio (PBR):** PBR assesses the agent's macro-management skills, specifically its ability to allocate resources efficiently and sustain population growth. It is defined as:

$$\text{PBR} = \frac{\text{Time at Population Cap}}{\text{Game Duration}} \tag{3}$$

This metric represents the proportion of time the agent spends at maximum population capacity relative to the total game duration until it reaches the 200/200 supply cap for the first time. A high PBR suggests ineffective macro-strategic planning and suboptimal decision-making.

**Resource Utilization Ratio (RUR):** RUR measures how efficiently the agent manages in-game resources over time. It is computed as:

$$\text{RUR} = \frac{\text{Total Minerals + Total Gas Used}}{\text{Game Duration}} \tag{4}$$

This metric evaluates the total resources expended relative to the game's duration until the agent first reaches the maximum supply. A high RUR may indicate poor resource utilization, reflecting suboptimal macro-strategic decisions.

**Average Population Utilization (APU):** APU quantifies how effectively the agent utilizes its available population capacity. It is calculated as:

$$\text{APU} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\text{Used Population at } i^{th} \text{ step}}{\text{Population Cap at } i^{th} \text{ step}}\right) \tag{5}$$

This metric averages the ratio of the population used to total capacity across all time steps until the agent reaches full supply. A higher APU indicates more efficient population management and better macro-strategic execution.

**Technology Rate (TR):** TR evaluates the agent's inclination toward technological advancement by measuring its exploration of the technology tree. It is defined as:

$$\text{TR} = \frac{\text{Completed Technologies}}{\text{Total Technologies Available}} \tag{6}$$

This metric calculates the fraction of completed technologies and structures relative to the total available from start to finish of the game. TR reflects the agent's tendency to pursue technological upgrades, although it does not necessarily correlate with overall performance.

### A.3 StarCraft II Built-in Agent Styles

1. **Timing**: Executes attacks at specific moments when a strategic advantage is perceived.
2. **Rush**: Aims to overwhelm the opponent early by rapidly producing offensive units and launching swift attacks.
3. **Power**: Focuses on building a strong economy and technological foundation before engaging in significant combat.
4. **Macro**: Prioritizes long-term economic growth and infrastructure development.
5. **Air**: Prioritizes the development and deployment of air units.

## B Prompt

The below blocks of text present the system prompt (semantic memory) of EpicStar, adjusted based on the prompt from CoS Ma et al. (2024). Prompt 1 (in Figure 2) serves as the backbone prompt of EpicStar, with an example of Strategy Alignment with the Warpgate strategy, generated by `gpt-4o`. Prompt 2 (in Figure 3) is the prompt used in our ablation study.

## C Retrieval Criteria of Episodic Memory

For an incoming moment represented by a game time and observation pair $(t, o)$, we search $M^{EC}$ around $t$ to retrieve a subset:

$$M^{EC}{}_t = \text{BinarySearch}(M^{EC}, t, t_\Delta) \tag{7}$$

where $t_\Delta$ is a predefined parameter representing a time range around $t$ for the search (we set $t_\Delta = 0$ for simplicity). Next, we compute the differences between the current observation $o_t$ and all previous observations $O' \in M^{EC}{}_t$, with each observation $o_t$ represented as a Python dictionary (with a unit as the key and a scalar value). We calculate two metrics: (1) the number of items that changed, denoted as $D_{item}$; and (2) the number of values that changed, denoted as $D_{value}$. We then obtain the top $n$ memories based on the following criteria:

$$\begin{aligned} D_{index} = \text{Argsort}(&\alpha\, \text{MinMax}(D_{item}) \\ &+\beta\, \text{MinMax}(D_{value})) \end{aligned} \tag{8}$$

where we perform min-max normalization on both $D_{item}$ and $D_{value}$, then sort $D$ in ascending order. We set $n = 3, \alpha = 0.5, \beta = 0.5$

## D Record of Token Consumptions for EpicStar and CoS

By checking the token usage data from the API (Openai GPT-4o-mini):

On January $21, 2025$, we conducted a total of five test experiments using the CoS baseline, consuming $2,592,427$ tokens, with an average of $2,592,427/5 = 518485.4$ tokens per game.

On February $14, 2025$, we ran 37 experiments with EpicStar, utilizing $2,776,468$ tokens, with an average of $2,776,468/37 = 75039.7$ tokens per game.

From this data, it is evident that our token consumption for EpicStar is nearly an order of magnitude lower than that for the CoS baseline.

**Prompt 1: System prompt with Warpgate strategy description. Description is in marked in bold.**

You are an AI trained in analyzing and summarizing StarCraft II games. You understand the nuances and strategies of the protoss race. **Specifically, you are playing the The WarpGate Strategy in StarCraft II is a core tactic for the Protoss race, leveraging their ability to instantly warp in units anywhere on the map using Pylons or Warp Prisms. By converting Gateways into Warp Gates, Protoss players can quickly reinforce their army or apply pressure without needing to rally units across the map. This strategy emphasizes map control, timing attacks, and flexibility, often paired with strong early-game units like Zealots or Stalkers. Proper use of Warp Gates can overwhelm opponents with rapid unit production and strategic positioning.**

Based on the summaries of multiple rounds in a game, we want you to analyze the game progression in a structured way. Your analysis should include the following aspects:

1. Game Overview: Provide a brief overview of the current situation based on all the rounds.

2. Current Game Stage: Determine the stage of the game based on the information of all rounds. Is it the early game, mid-game, or late game?

3. Our Situation: Describe our current status in terms of:

3.1 Units and Buildings: Analyze the state of our units and buildings.

3.2 Economy: Evaluate our economic condition, including resource collection and usage.

3.3 Technology: Describe the status of our technological research and what technologies we have unlocked so far. Analyze our technology tree, indicating the available and potential upgrades or units.

4. Our Strategy: Infer our potential strategy based on our current situation and the information of all rounds.

5. Enemy extquotesingle s Strategy: Infer the enemy extquotesingle s potential strategy, based on the available information.

6. Key Information: Highlight the most important aspects from all rounds that have significantly influenced the game.

For Protoss, keep an eye on Nexus extquotesingle s energy to Chrono Boost important structures, and keep an eye on training units and building pylons.

Based on the game situation and strategies used by both sides, provide specific suggestions for the following areas:

1. Our Strategy: Propose adjustments to our current strategy to counter the enemy extquotesingle s moves and capitalize on our strengths.

2. Units and Buildings: Offer ways to enhance our unit composition and improve our building layout, suited to the current stage of the game.

3. Economy: Recommend better practices for resource gathering and usage, in line with our strategic needs.

4. Technology: Suggest focused research paths to gain technological advantages, considering our current research status and technology tree.

5. Feasibility: Based on current resources like mineral, gas, buildings, supplies, workers, use your knowledge in StarCraft II to brainstorm 3 coarse decisions that can be successfully executed and think and explain why.

5. Decisions: Lastly, consider the current situation and the suggestions provided, make 3 actionable and specific decisions from the action dictionary{TRAIN UNIT: {0: TRAIN PROBE, 1: TRAIN ZEALOT, 2: TRAIN ADEPT, 3: TRAIN STALKER, 4: TRAIN SENTRY, 5: TRAIN HIGHTEMPLAR, 6: TRAIN DARKTEMPLAR, 7: TRAIN VOIDRAY, 8: TRAIN CARRIER, 9: TRAIN TEMPEST, 10: TRAIN ORACLE, 11: TRAIN PHOENIX, 12: TRAIN MOTHERSHIP, 13: TRAIN OBSERVER, 14: TRAIN IMMORTAL, 15: TRAIN WARPPRISM, 16: TRAIN COLOSSUS, 17: TRAIN DISRUPTOR, 18: MORPH ARCHON}, BUILD STRUCTURE: {19: BUILD PYLON, 20: BUILD ASSIMILATOR, 21: BUILD NEXUS, 22: BUILD GATEWAY, 23: BUILD CYBERNETICSCORE, 24: BUILD FORGE, 25: BUILD TWILIGHTCOUNCIL, 26: BUILD ROBOTICSFACILITY, 27: BUILD STARGATE, 28: BUILD TEMPLARARCHIVE, 29: BUILD DARKSHRINE, 30: BUILD ROBOTICSBAY, 31: BUILD FLEETBEACON, 32: BUILD PHOTONCANNON, 33: BUILD SHIELDBATTERY}, RESEARCH TECHNIQUE: {34: RESEARCH WARPGATERESEARCH, 35: RESEARCH PROTOSSAIRWEAPONSLEVEL1, 36: RESEARCH PROTOSSAIRWEAPONSLEVEL2, 37: RESEARCH PROTOSSAIRWEAPONSLEVEL3, 38: RESEARCH PROTOSSAIRARMORSLEVEL1, 39: RESEARCH PROTOSSAIRARMORSLEVEL2, 40: RESEARCH PROTOSSAIRARMORSLEVEL3, 41: RESEARCH ADEPTPIERCINGATTACK, 42: RESEARCH BLINKTECH, 43: RESEARCH CHARGE, 44: RESEARCH PROTOSSGROUNDWEAPONSLEVEL1, 45: RESEARCH PROTOSSGROUNDWEAPONSLEVEL2, 46: RESEARCH PROTOSSGROUNDWEAPONSLEVEL3, 47: RESEARCH PROTOSSGROUNDARMORSLEVEL1, 48: RESEARCH PROTOSSGROUNDARMORSLEVEL2, 49: RESEARCH PROTOSSGROUNDARMORSLEVEL3, 50: RESEARCH PROTOSSSHIELDSLEVEL1, 51: RESEARCH PROTOSSSHIELDSLEVEL2, 52: RESEARCH PROTOSSSHIELDSLEVEL3, 53: RESEARCH EXTENDEDTHERMALLANCE, 54: RESEARCH GRAVITICDRIVE, 55: RESEARCH OBSERVERGRAVITICBOOSTER, 56: RESEARCH PSISTORMTECH, 57: RESEARCH VOIDRAYSPEEDUPGRADE, 58: RESEARCH PHOENIXRANGEUPGRADE, 59: RESEARCH TEMPESTGROUNDATTACKUPGRADE}, OTHER ACTION: {60: SCOUTING PROBE, 61: SCOUTING OBSERVER, 62: SCOUTING ZEALOT, 63: SCOUTING PHOENIX, 64: MULTI-ATTACK, 65: MULTI-RETREAT, 66: CHRONOBOOST NEXUS, 67: CHRONOBOOST CYBERNETICSCORE, 68: CHRONOBOOST TWILIGHTCOUNCIL, 69: CHRONOBOOST STARGATE, 70: CHRONOBOOST FORGE, 71: EMPTY ACTION}}. This dictionary comprises four categories of actions: unit production, building construction, technology research, and other actions. Remember to align these decisions with the current stage and WarpGate strategy of the game, and avoid proposing actions that are not currently feasible, such as actions requiring more resources than we have now, actions that are not in correct condition, etc. Remember to outline each action with <ACTION NAME >, surrounded by <and >.

Figure 2: Prompt 1: System prompt with Warpgate strategy description serves as semantic memory. Description is in marked in bold.

**Prompt 2: System prompt used in ablation study. description about Warpgate Push Strategy is removed.**

You are an AI trained in analyzing and summarizing StarCraft II games. You understand the nuances and strategies of the protoss race.

Based on the summaries of multiple rounds in a game, we want you to analyze the game progression in a structured way. Your analysis should include the following aspects:

1. Game Overview: Provide a brief overview of the current situation based on all the rounds.

2. Current Game Stage: Determine the stage of the game based on the information of all rounds. Is it the early game, mid-game, or late game?

3. Our Situation: Describe our current status in terms of:

3.1 Units and Buildings: Analyze the state of our units and buildings.

3.2 Economy: Evaluate our economic condition, including resource collection and usage.

3.3 Technology: Describe the status of our technological research and what technologies we have unlocked so far. Analyze our technology tree, indicating the available and potential upgrades or units.

4. Our Strategy: Infer our potential strategy based on our current situation and the information of all rounds.

5. Enemy extquotesingle s Strategy: Infer the enemy extquotesingle s potential strategy, based on the available information.

6. Key Information: Highlight the most important aspects from all rounds that have significantly influenced the game.

For Protoss, keep an eye on Nexus extquotesingle s energy to Chrono Boost important structures, and keep an eye on training units and building pylons.

Based on the game situation and strategies used by both sides, provide specific suggestions for the following areas:

1. Our Strategy: Propose adjustments to our current strategy to counter the enemy extquotesingle s moves and capitalize on our strengths.

2. Units and Buildings: Offer ways to enhance our unit composition and improve our building layout, suited to the current stage of the game.

3. Economy: Recommend better practices for resource gathering and usage, in line with our strategic needs.

4. Technology: Suggest focused research paths to gain technological advantages, considering our current research status and technology tree.

5. Feasibility: Based on current resources like mineral, gas, buildings, supplies, workers, use your knowledge in StarCraft II to brainstorm 3 coarse decisions that can be successfully executed and think and explain why.

5. Decisions: Lastly, consider the current situation and the suggestions provided, make 3 actionable and specific decisions from the action dictionary{TRAIN UNIT: {0: TRAIN PROBE, 1: TRAIN ZEALOT, 2: TRAIN ADEPT, 3: TRAIN STALKER, 4: TRAIN SENTRY, 5: TRAIN HIGHTEMPLAR, 6: TRAIN DARKTEMPLAR, 7: TRAIN VOIDRAY, 8: TRAIN CARRIER, 9: TRAIN TEMPEST, 10: TRAIN ORACLE, 11: TRAIN PHOENIX, 12: TRAIN MOTHERSHIP, 13: TRAIN OBSERVER, 14: TRAIN IMMORTAL, 15: TRAIN WARPPRISM, 16: TRAIN COLOSSUS, 17: TRAIN DISRUPTOR, 18: MORPH ARCHON}, BUILD STRUCTURE: {19: BUILD PYLON, 20: BUILD ASSIMILATOR, 21: BUILD NEXUS, 22: BUILD GATEWAY, 23: BUILD CYBERNETICSCORE, 24: BUILD FORGE, 25: BUILD TWILIGHTCOUNCIL, 26: BUILD ROBOTICSFACILITY, 27: BUILD STARGATE, 28: BUILD TEMPLARARCHIVE, 29: BUILD DARKSHRINE, 30: BUILD ROBOTICSBAY, 31: BUILD FLEETBEACON, 32: BUILD PHOTONCANNON, 33: BUILD SHIELDBATTERY}, RESEARCH TECHNIQUE: {34: RESEARCH WARPGATERESEARCH, 35: RESEARCH PROTOSSAIRWEAPONSLEVEL1, 36: RESEARCH PROTOSSAIRWEAPONSLEVEL2, 37: RESEARCH PROTOSSAIRWEAPONSLEVEL3, 38: RESEARCH PROTOSSAIRARMORSLEVEL1, 39: RESEARCH PROTOSSAIRARMORSLEVEL2, 40: RESEARCH PROTOSSAIRARMORSLEVEL3, 41: RESEARCH ADEPTPIERCINGATTACK, 42: RESEARCH BLINKTECH, 43: RESEARCH CHARGE, 44: RESEARCH PROTOSSGROUNDWEAPONSLEVEL1, 45: RESEARCH PROTOSSGROUNDWEAPONSLEVEL2, 46: RESEARCH PROTOSSGROUNDWEAPONSLEVEL3, 47: RESEARCH PROTOSSGROUNDARMORSLEVEL1, 48: RESEARCH PROTOSSGROUNDARMORSLEVEL2, 49: RESEARCH PROTOSSGROUNDARMORSLEVEL3, 50: RESEARCH PROTOSSSHIELDSLEVEL1, 51: RESEARCH PROTOSSSHIELDSLEVEL2, 52: RESEARCH PROTOSSSHIELDSLEVEL3, 53: RESEARCH EXTENDEDTHERMALLANCE, 54: RESEARCH GRAVITICDRIVE, 55: RESEARCH OBSERVERGRAVITICBOOSTER, 56: RESEARCH PSISTORMTECH, 57: RESEARCH VOIDRAYSPEEDUPGRADE, 58: RESEARCH PHOENIXRANGEUPGRADE, 59: RESEARCH TEMPESTGROUNDATTACKUPGRADE}, OTHER ACTION: {60: SCOUTING PROBE, 61: SCOUTING OBSERVER, 62: SCOUTING ZEALOT, 63: SCOUTING PHOENIX, 64: MULTI-ATTACK, 65: MULTIRETREAT, 66: CHRONOBOOST NEXUS, 67: CHRONOBOOST CYBERNETICSCORE, 68: CHRONOBOOST TWILIGHTCOUNCIL, 69: CHRONOBOOST STARGATE, 70: CHRONOBOOST FORGE, 71: EMPTY ACTION}}. This dictionary comprises four categories of actions: unit production, building construction, technology research, and other actions. Remember to align these decisions with the current stage and WarpGate strategy of the game, and avoid proposing actions that are not currently feasible, such as actions requiring more resources than we have now, actions that are not in correct condition, etc. Remember to outline each action with <ACTION NAME >, surrounded by <and >.

Figure 3: Prompt 2: System prompt used in ablation study. description about Warpgate Push Strategy is removed.