AimBot
An investigation into external memory and advanced prompting

Author - Mehul Maheswari)
mmaheshwari@ucsd.edu

1 Introduction

1.1 Problem Definition

Large Language Models (LLMs) have trended towards increasing parameter sizes to improve their performance across a wide range of tasks. However, recent advances demonstrate that smaller-scale models, when combined with knowledge bases (KBs) and effective prompting strategies, can achieve comparable results while being more efficient and interpretable (Borgeaud et al., 2022). This project investigates strategies for integrating KBs and optimized prompting techniques in generating high-quality, task-specific responses.

1.2 Project Goals and Outcomes

- **Collect and pre-process data for an external memory solution. DONE.** Successfully curated a KB in JSON format and ensured compatibility with the agent's querying pipeline.
- **Investigate effectively transferring relevant information from the KB to the LLM: DONE.** Implemented and validated strategies to ensure seamless data access and retrieval for the agent.
- **Design and optimize prompts to guide the agent in generating high-quality, contextually accurate responses: DONE.** Explored and refined prompting techniques to enhance the clarity and relevance of the model's outputs.

2 Related work

The growth and popularity of large language models (LLMs) has been shadowed by significant research into their scaling, usages, and enhancement through external memory systems and prompting techniques.

A critical aspect of LLMs' utility lies in their size and capability scaling. Brown et al. (2020) introduced GPT-3, showcasing how increasing model parameters can lead to a dramatic improvement in performance across many tasks, establishing the importance of scale in LLM development (Brown et al., 2020).

However soon afterwards, incorporating external memory into LLM architecture was shown to achieve similar results. Borgeaud et al. (2022) introduced RETRO, a retrieval-augmented model demonstrating how access to a vast external knowledge base can reduce model size while maintaining high-quality output (Borgeaud et al., 2022). This approach emphasized the role of memory systems in scaling models effectively without prohibitive computational costs from exploding hyper-parameters.

Simultaneously, the exploration of prompting strategies demonstrated that high quality responses can be generated outside of increased parameters. Chain-of-Thought (CoT) prompting has proven particularly effective for improving multi-step reasoning, as detailed by Wang et al. (2023), who provided an empirical study highlighting CoT's ability to elicit structured reasoning paths from models (Wang et al., 2023).

The effectiveness of prompting depends heavily on its design. A systematic survey by Wei et al. (2022) outlined strategies such as zero-shot and few-shot prompting, emphasizing how high-quality prompts directly influence LLM performance on complex tasks (Wei et al., 2022). This work demonstrates the need for thoughtful prompt engineering to maximize task-specific results.

Finally, optimizing memory systems and retrieval mechanisms remains critical. A study by Wu et al. (2023) presented techniques for aligning external memory usage with task requirements, ensuring that models efficiently access relevant in-

formation while avoiding unnecessary computational overhead (Wu et al., 2023).

Collectively, these studies show that LLM performance is not solely a function of model size, and that intelligent information retrieval and processing can increase performance. This project builds upon this foundation, and explores how to implement these strategies to optimize model performance.

3 Dataset and Preprocessing

Valorant players present a unique dataset characterized by dynamic player performance metrics. Unlike traditional sports with more standardized statistics, Valorant's competitive landscape requires more nuanced statistical interpretation, making it a great testing ground for advanced knowledge base integration strategies.

3.1 Raw Data Overview

The initial dataset consisted of log files from three major tournaments: VCT Challengers, Game Changers, and VCT International. Across these, I collected a total of 7,357 game files, amounting to 101.1 GB of data. Each game file consisted of a series of JSON events capturing different aspects of gameplay. For instance, the initialization event from a log file is shown below:

Listing 1: Example Game Log Event

```
{
    "platformGameId": "val:004b09b1-4dc
        9-4185-baff-9b1c66b3ef99",
    "playerDied": {
        "deceasedId": { "value": 2 },
        "weapon": {
            "fallback": {
                "displayName": "",
                "inventorySlot": { "
                    slot": "PRIMARY" },
                "guid": "EE8E8D15-496B-
                    07AC-E5F6-8FAE5D4C7B
                    1A"
            },
            "type": "UNKNOWN"
        },
        "assistants": [
            { "assistantId": { "value":
                7 } }
        ],
        "killerId": { "value": 6 }
    }
}
```

3.2 Data Preprocessing

Given the sheer volume and complexity of the raw data, direct ingestion into a knowledge base was far too expensive, and realistically infeasible. The preprocessing pipeline was designed to achieve the following:

1. **Identify Relevant Information:** Extract key metrics from the logs, such as kills, deaths, assists, combat score, and other performance-related data.

2. **Aggregate Player Data:** Organize information across multiple files to account for players participating in multiple games. I used vlr.gg as a reference for how to structure this information.

3. **Format for Knowledge Base:** Represent each player's aggregated performance as a JSON object to facilitate fast querying and compatibility with the knowledge base's indexing system.

Example Processed Entry An example of the processed data is shown below. It encapsulates a player's performance metrics, both overall and segmented by the agent roles they played:

Listing 2: Processed Player Data

```
{
    "handle": "Stefanie",
    "date": "2023-10-03T08:46:46",
    "status": "active",
    "first_name": "Stefanie",
    "last_name": "Jones",
    "home_team_name": "Version 1",
    "home_team_acronym": "V1",
    "career_statistics": {
        "total_rounds_played": 157,
        "attack_kda": 0.84,
        "defense_kda": 1.48,
        ...
    },
    "player_statistics_per_agent": {
        "Sentinel": {
            "Killjoy": {
                "total_rounds_played":
                    115,
                "attack_kda": 0.71,
                ...
            },
            ...
        }
    }
}
```

3.3 Rationale for JSON Representation

The decision to represent data as JSON objects per player was driven by the requirements of the knowledge base and the evaluation metrics. This structure allows for efficient lookups of specific statistics, such as a player's Kill-Death Ratio

(KDA) or average combat score. Additionally, the format aligns with the focus on statistical evaluation metrics, ensuring compatibility and flexibility during knowledge base queries.

4 Evaluation Metrics

To evaluate the accuracy and reliability of responses generated by the model, I employed the following baselines:

4.1 Statistics Match Score

The *Statistics Match Score* determines whether the data referenced in the model's response exists in the knowledge base or corresponds to real-world information. This metric is designed to get a quick understanding of the models overall performance accounting for hallucinations, or where the model fabricates and misrepresents statistical data. Responses are assigned a score of 0 if they lack statistical references or contain hallucinated data, and a score of 1 if the referenced statistics are verified as accurate and present or not.

We check the validity of the information through vlr.gg, an online platform that contains the most recent statistics for Valorant players. The information from vlr.gg is also how the knowledge base's statistics are modeled, which helps standardize performance evaluation.

4.2 Error Classification

To gain deeper insight into incorrect responses, I classify errors into the following categories:

1. **Hallucination:** The statistic is fabricated and has no basis in the knowledge base or real-world data.

2. **Omission:** The response does not include statistics when they are contextually required.

3. **Context Mismatch:** The statistic exists, but is irrelevant or inaccurately applied within the response. For example, if one players statistics get used as a response for another.

By analyzing responses through this framework, we can identify failure patterns and inform the development of the knowledge base and prompting strategies.

4.3 Response Latency

In addition to error classification metrics, I measured *Latency*, which is the time taken by the model to generate an answer on average. This is is a practical consideration for real-world application, and although not crucial, not negligible.

PS: Note that this investigation and method did not require the need for a test/train/validation split or hyper-parameter tuning.

5 My Approach: Models

5.1 Model 0: Baseline

I chose AWS Bedrock as the platform for this study due to its integrated knowledge base capabilities, which streamline the investigation process. Additionally, its "traces" feature provides insights into the LLM's decision-making process before generating a response, which would be useful for analysis. Furthermore, because I chose to use an existing third party service, I did not have to worry about computational constraints, aside from the storage for embedding representations with the external memory data sources.

Among the available models on AWS Bedrock, I selected Anthropic's Claude 3 Sonnet for its state-of-the-art capabilities. The initial prompt I used was:

> "You are a helpful agent that gives responses about Valorant teams and players. Include statistics in your responses."

This prompt adheres to prompting conventions, being concise, straightforward, and directed on the evaluation metrics, which are heavily statistic-oriented.

Example Query and Response:

> Q: "Build me a team of five players" R: "For a team of 5 Valorant players focused on aggressive playstyles, here is my recommendation: Duelist: TenZ (NA) - One of the best aimers and Jett players. Known for his aggressive entries and clutch plays. Duelist: ScreaM (EU) - Veteran FPS player with incredible aim and movement. Plays agents like Raze and Jett. Initiator: Sinatraa (NA) - Former Overwatch pro with great mechanics and game sense. Plays Sova and other initiators. Controller: Hiko (NA) - Experienced in-game leader and strategic controller player. Known for his Viper play. Sentinel: Dapr (NA) -

Solid sentinel player who supports his teammates well with utility and clutch plays."

Statistics Match Score: 2.5	
Hallucinations: 5	**Omissions: 15**
Context Mismatches: 0	**Latency: 24.08s**

Table 1: Results over 20 Queries

As shown in table 2, the baseline model often failed to root its recommendations in statistical data. In the few instances statistics were provided, they were often outdated or simply incorrect.

5.2 Model 1: Initial Knowledge Base Integration

5.2.1 Approach

The primary limitation of the baseline model was the frequent omission of statistics, a critical shortfall for the intended application. To address this, I incorporated an external memory resource into the architecture, leveraging Amazon Titan Text Embedding v2 for its superior performance in embedding and optimizing textual data for retrieval tasks (Amazon Web Services, 2024).

The knowledge base employs a K-Nearest Neighbors (KNN) retrieval mechanism facilitated by Faiss, a library renowned for its efficiency in vector search and clustering (Project, 2024; Research, 2024). Given the JSON-based structure of the external memory, I set a token limit of 300 for chunk-based retrieval, a parameter chosen to balance the trade-offs between retrieval granularity and response latency.

5.2.2 Results

Example KB Response:

"Sentinel": {"Killjoy": {"total_rounds_played": 47, "attack_kda": 1.0, "defense_kda": 0.83, "avg_kills_per_round": 0.53, "avg_assists_per_round": 0.13, "avg_combat_score_per_round": 175.74, "avg_revives_per_round": 0.0, "avg_damage_dealt_per_round": 105.83, "avg_first_bloods_per_round": 0.15, "avg_first_deaths_per_round": 0.09}}, {"handle": "Wo0t", "date": "2024-04-10T08:29:38", "status": "active", "first_name": "Mert", "last_name": "Alkan", "home_team_name": "Team Heretics", "home_team_acronym": "TH"}

Statistics Match Score: 14	
Hallucinations: 0	**Omissions: 6**
Context Mismatches: 8	**Latency: 16.45s**

Table 2: Model 1 results over 20 queries

5.2.3 Analysis

The results from table 2 are polarizing because they demonstrate both the strengths and challenges of the knowledge base retrieval. The inclusion of a knowledge base significantly improved statistical retrieval as demonstrated by the reduced hallucinations, and increased statistics match score. However, several challenges remain:

1. **Chunking Issues:** The 300-token chunking strategy frequently truncates JSON objects, leading to incomplete or clipped statistics. This undermines the model's ability to provide accurate responses for larger player profiles.

2. **Similarity Strategy:** The reliance on cosine similarity for vector-based retrieval often skews the selection towards highly specific segments. This approach neglects the broader contextual coherence required for responses involving multiple interrelated data points.

3. **Dynamic JSON Sizes:** JSON size varies based on the number of games played by each player, making fixed-size chunking unreliable.

Another concern that was demonstrated in the responses but not captured by the metrics was the semantic variation in the answer that the LLM produced. Although statistics were often included, how they were used ranged quite a bit. Ideally, we would want to reduce this variation to ensure a more defined path for searching statistics in our knowledge base.

5.3 Model 2: Revised Knowledge Base Integration and Advanced Prompting Strategies

5.3.1 Approach

Building on the prior model, we can see avenues for improvement using better chunking strategies

to ensure better statistic retrieval as well as opportunities for more consistency in knowledge base lookup and answer generation.

To address the issue of contextually mismatched statistics in Model 1, a more refined chunking strategy is introduced. I used a hierarchical chunking approach which is designed to preserve the integrity of the data while reducing the likelihood of fragmented player statistics. This strategy uses a two-tier system with a parent chunk size of 1500 tokens and a child chunk size of 300 tokens. The parent chunks encapsulate broader contexts, while the child chunks break down the detailed statistics for individual players. This segmentation allows for more accurate retrieval of player data, ensuring that relevant statistics are retained in their entirety and improving the overall accuracy of the model's responses.

In addition to the chunking strategy, I drastically revised the prompting strategy to improve the consistency and accuracy of the responses generated by the model. The new prompt is based on the prompting guidelines from (Anthropic, 2024), which provide detailed strategies for working with Claude models. The revised prompt integrates advanced prompting techniques, including chain of thought reasoning and XML tagging, to guide the model. The "chain of thought" method encourages the model to follow a more consistent and structured logical flow when reasoning, while XML tags are used to mark key elements within the response, helping the model focus on specific pieces of information from the player statistics.

5.3.2 Results

Statistics Match Score: 14.5	
Hallucinations: 0	**Omissions: 0**
Context Mismatches: 6	**Latency: 45.08s**

Table 3: Model 2 results over 20 queries

5.3.3 Analysis

As demonstrated by table 3, the model improved in consistently queering statistics with zero omissions or hallucinations but still had some context mismatches. Also, the response time increased considerably.

One of the main hurdles in improving this model is designing a prompt that is both specific and effective. The prompt needs to be clear and concise, while detailing more complex tasks such as using the knowledge base to retrieve statistics. As we can see, including CoT prompting comes with a large response time increase, so finding an optimific prompt requires some tenacity. On the bright side, instances when statistical information was omitted never occurred, suggesting the prompting helped structure the way the LLM responds to queries.

Furthermore, despite the introduction of the hierarchical clustering mechanism, the chunking strategy still does not capture the full range of the player JSON, clipping irrelevant player information from one player with the other. This issue contributed to context mismatches, which we can see was a prevalent problem for this model.

5.4 Model 3: Knowledge Base with new Chunking Strategy and Revised Prompting

5.4.1 Approach

Although the last model was a significant step forward from the first model and baseline, revisions to both the chunking and prompting strategies can lead to significant improvement.

For this model we implement a custom chunking strategy by spitting up each players data into a separate file, and then feeding the separate documents for query processing. Instead of working with chunks, the system now queries documents which ensures player information is complete and accurate. This strategy absolves the need for a chunking mechanism along with the clipping problem prior iterations have faced.

5.4.2 Results

Example question and response from model 3:

> Q: Build a team of 5 players from VCT challengers A: Here is my proposed 5-player team from VCT Challengers: - Duelist: Joseph "clear" Allen (Raze, Jett) - Excels at getting opening kills with 0.19 first bloods per round on Raze - High 1.38 attack KD and 236.77 combat score per round on Raze - Controller: Matias "vaiZ" Matias (Brimstone, Harbor, Omen) - Versatile controller with strong stats across multiple agents - 1.87 attack KD and 227.8 combat score per round on Brimstone - Initiator: Min-Seung "SID" Kim (Gekko, Sova) - Deadly with initiator agents like

Gekko (1.73 attack KD) - High 293.53 combat score per round on new agent ISO - Sentinel: shao yi qun "Nicc" (Killjoy, Chamber) - Consistent performer on Killjoy with 1.44 attack KD - Impacts the game through utility usage with high assists per round - Sentinel: Matias "khz" Bugueño (Killjoy, Raze) - Versatile and can play both sentinel and duelist roles - Standout 1.64 defense KD and 246.97 combat score on Raze

Statistics Match Score: 18	
Hallucinations: 0	**Omissions: 2**
Context Mismatches: 0	**Latency: 31.12s**

Table 4: Results over 20 Queries

5.4.3 Analysis

As we can see, the custom chunking strategy has succeed as the model is able to accurate assign information to each player. By ensuring that the chunks are what we intended them to be, we have taken a major step forward compared to the previous model which tried to make sense of fragmented information.

Alongside the custom chunking strategy, we also refined the prompting approach. In the prior model, a separate prompt was included in the knowledge base to instruct how the queried information should be processed. Although necessary then, in this model the additional prompt proved to be detrimental, as it interfered with the model's ability to retrieve and utilize the correct data. The inclusion of this prompt caused incomplete traces, likely due to its complex nature, which conflicted with the clarity needed for effective chain of thought prompting. Upon removing this extra layer of instruction, the model's responses regained their accuracy and coherency.

As a result of these changes, this model is not only much faster than previous models, but it also shows improved accuracy, especially for more specific, less open-ended queries. For example, questions about specific players are now handled with higher precision, yielding responses that are consistent and highly relevant.

6 Conclusion

This project investigated integrating external knowledge bases and prompting techniques with Large Language Models (LLMs) to improve response accuracy and reduce hallucinations. My findings reveal the following insights:

6.0.1 Knowledge Base integration

Model iterations demonstrated the impact of knowledge base inclusion on reducing hallucinations. Starting from the baseline model, we developed an approach that effectively minimized incorrect information generation. The evolution from the baseline to Model 3 showed a dramatic improvement in the Statistics Match Score, rising from 2.5 to 18 across 20 queries.

Our study found that the chunking strategy used for to query the knowledge base is imperative and must be considered from the standpoint of the data source being used. Faulty implementations can result in incorrect or mismatched responses, along with higher computational complexity if implemented with chain of thought prompting.

6.0.2 Prompting

Prompting was a crucial mechanism for structuring responses from the LLM, as well as utilizing information from the knowledge base. The research across models showed the following insights:

- Overly complex prompts can hinder model performance
- Chain of Thought (CoT) prompting can improve response structure
- The complexity of prompting directly impacts response time and accuracy

6.0.3 Future Work

Future work may include the following directions:

- Exploring alternative embedding strategies to improve chunking mechanisms
- Exploring lightweight prompting techniques to reduce computational overhead
- Explore alternative chunking mechanisms

7 Acknowledgements

For all sections, ChatGPT was used to proofread and format as appropriate.

References

Amazon Web Services (2024). Titan embedding models - amazon bedrock. https://docs.aws.amazon.com/bedrock/latest/userguide/titan-embedding-models.html. Accessed: 2024-12-03.

Anthropic (2024). Prompt engineering with claude. Accessed: 2024-12-03.

Borgeaud, S., Mensch, A., Hoffmann, J., et al. (2022). Improving language models by retrieving from trillions of tokens. In *Advances in Neural Information Processing Systems*.

Brown, T., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Project, O. (2024). K-nearest neighbors (k-nn) index. Accessed: 2024-12-03.

Research, F. A. (2024). Faiss wiki. Accessed: 2024-12-03.

Wang, B., Min, S., Deng, X., et al. (2023). Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Wei, J. et al. (2022). A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Wu, Z. et al. (2023). Optimizing external memory in neural language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.