Utilising Web-Scraped Steam Data: Retrieval-Augmented Generation Implementation on Large Language Models

MSIN0166: Data Engineering
Individual Coursework

Word Count: 3753

Github Repository Link :

https://github.com/YohanesNelsen/DataEngIndividual

(this is a private repository, only user provided in moodle can access this link)

University College London

Master Science of Business Analytics

Table of Content

1. Introduction

In the rapidly evolving field of artificial intelligence, Large Language Models (LLMs) such as GPT-4 have revolutionised how businesses interact with data. LLMs can generate coherent, context-aware text based on vast amounts of training data, making them indispensable in various applications ranging from automated customer service to data processing. Despite their prowess, these models are not without limitations; one significant challenge is their tendency to generate information that, while plausible, may be factually incorrect—a phenomenon known as "hallucination." According to Harvard Business Review, 79% of senior IT leaders reported concerns that these technologies (GenAI) bring the potential for security risks, and another 73% are concerned about biassed outcomes (Baxter et al.; Y., 2023).

To overcome the limitations of Language Models, a new approach called Retrieval-Augmented Generation (RAG) can be applied. RAGs combine the power of Large Language Models (LLMs) with a retrieval mechanism that sources relevant information from a database of knowledge (Zendata, 2024). This hybrid technique allows the model to access relevant documents while generating responses, which helps to ground its output in authoritative content and reduces the risk of inaccuracies.

The choice to use data from Steam, the premier online platform for PC gaming, for this project is strategic. Steam's extensive database offers a rich source of real-time, user-generated content, encompassing game statistics, user reviews, and community interactions. This breadth of data not only provides diverse training material for LLMs but also presents a unique opportunity to explore consumer behaviour and preferences in the digital entertainment industry.

1.1 Business Integration

Implementing Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) in a business context provides organisations with advanced tools for decision-making and information retrieval. These technologies enable businesses to process and analyse large datasets efficiently, extracting valuable insights that can inform strategic planning and operational improvements. By utilising RAG, companies can enhance the accuracy and relevancy of the

information generated by LLMs, ensuring that decision-makers have access to reliable and contextually appropriate data. Not only that but by leveraging these technologies, businesses can also provide highly personalised responses and content recommendations based on the vast array of data points Steam collects. This approach enhances customer engagement and drives strategic marketing and product development initiatives based on nuanced data analysis.

2. Methodology

The methodology of this project involves a structured approach to collecting, cleaning, and analysing data from Steam, utilising advanced technologies to support the application of Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG). The process is divided into three main stages:

2.1 Data Collection through Web Scraping

Selenium and Edge WebDriver were used to gather comprehensive data from Steam. Selenium was chosen for its robust capabilities in automating web browsers, allowing for the intricate manipulation of webpage elements compared to Beautiful Soup. It is also capable of handling dynamic content when scraping (Udofia, 2024). This approach allows automated retrieval of diverse datasets directly from the website. The specific types of data collected include:

- **Pricing Data**: Prices of games across various countries are collected and converted to British pounds to standardise comparisons.
- **Review and Playhours Data**: Data on user reviews and gameplay hours are extracted to assess game popularity and user engagement.
- **General Game Information**: Data on player counts, developers, in-game activities, and classifications such as tags and categories, providing a holistic view of game features and market positioning.

2.2 Data Cleaning and Integration

Once the data is collected, it undergoes a cleaning and integration process using Apache Spark. This powerful platform is chosen for its ability to handle large datasets with high

scalability (Zahara et al., 2016). The cleaned data is then structured into three main Parquet files. Apache Spark's robust framework is particularly well-suited for this task due to its advanced data processing capabilities and its efficiency in handling large volumes of data, which is essential for the breadth of data extracted from Steam.

2.3 Implementation of RAG and LLM

Retrieval-Augmented Generation (RAG) was utilised with LLMs to enhance the output's reliability and relevance. The process involves generating embeddings for the textual data, which help in measuring the similarity between a user's query and the information chunks stored in the database. After that, LLM API provided at 'https://llm.api.ai8.io/query_llm' was used, The API aids in dynamically retrieving the most relevant data excerpts based on the embeddings, allowing the LLM to produce precise and contextually aware responses.

3. Pipeline

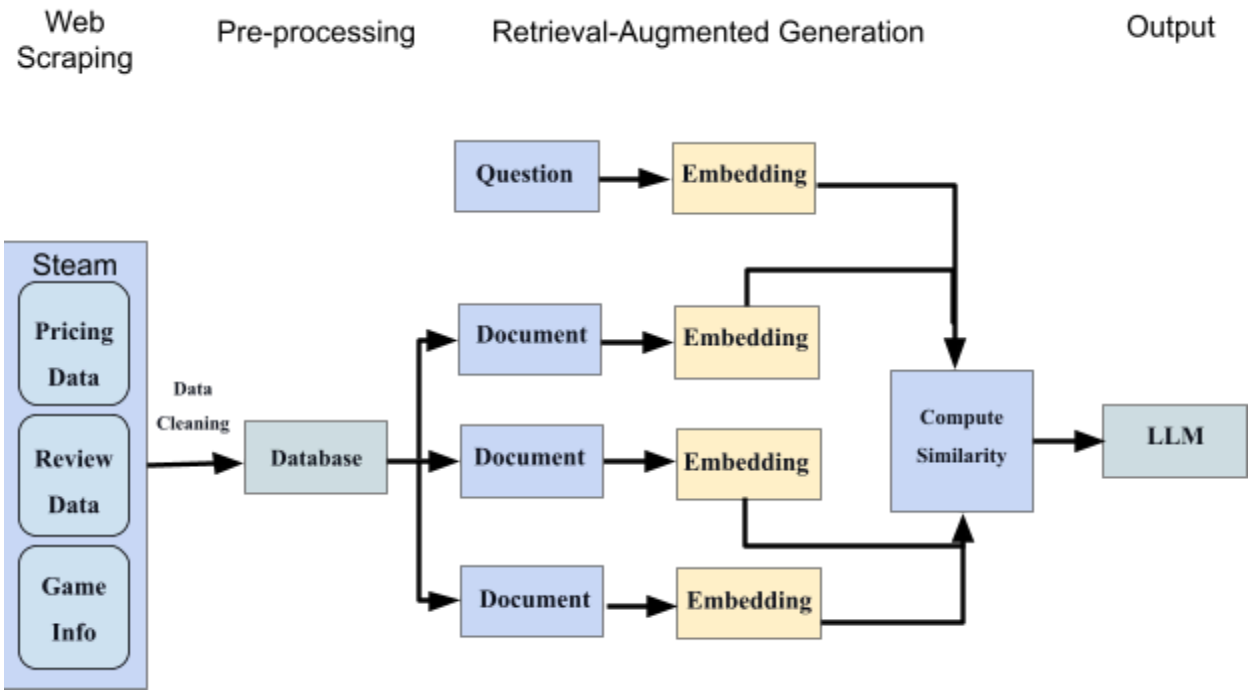

Figure 1: Project Pipeline

This project has four main stages: data collection, data cleaning, RAG implementation and LLM. This project was done in Python, and Apache Spark was also implemented to ensure the project was scalable to big data.

Web Scraping: Data collection phase where specific types of data are gathered from the web, in this case, from Steam. The data includes:

- Pricing Data contains data about each game's price from every country, the highest and lowest, and the converted price to pounds.
- Review Data contains review data, play hours and more for three games.
- Game Information contains general information about each game, such as tags, categories, and also player count

Pre-processing: Following data collection, the data undergoes pre-processing, which includes:

- Data Cleaning: This step is crucial to remove any errors, inconsistencies, or irrelevant data that could affect the quality of the output.
- The result then will be put into a database for further processing.

Retrieval-Augmented Generation: This is a sophisticated phase where the system enhances the generation process by using retrieved documents.

- Question: A query or question is proposed to the system.
- Embedding: Both the question and the various documents undergo an embedding process, transforming the text into a numerical form that can be processed computationally.
- Compute Similarity: The embedded question is compared to the document embeddings to find the most relevant information.

Output:

- LLM (Language Learning Model): The selected source with the highest similarity score to the question is then passed to a language learning model.
- The LLM then generates an output based on the input provided by the similarity computation.

The retrieval-augmented generation phase is central to this pipeline. It ensures that the output generated by the LLM is not only based on the model's pre-trained knowledge but also on specific information retrieved from the Steam data, making it more relevant and accurate.

4. Data Preprocessing

4.1 Web Scraping

A web scraping method using Selenium was used to ensure the data collected was up-to-date. The Microsoft Edge bot driver called 'edge driver' was used to automate the process and to ensure real-time accuracy and relevance. The process involved scraping data from two key websites: Steam to extract user reviews and play hours and SteamDB to gather pricing and general game information. The scraping technique was created to associate each piece of data with a specific game, utilising unique game IDs. Because the data was collected from up-to-date sources, the data will be updated each time the code runs. For this project's scope, the data was limited to approximately 150 reviews per game to optimise the operational efficiency without compromising the depth of insights. However, the code is scalable and can run to accommodate a larger volume of data or an increased number of games, should the need arise. Below is a table showcasing a glimpse of the data obtained from data collection.

Pricing	Review	General Information
Game	Game	Title
Currency (Country Name)	ReviewText	Developer
Current Price	Review (Recommended or Not Recommended)	In-Game Count
Converted Price	Review Length	Tags
Lowest Recorded Price	Play Hours	Categories
	Date Posted	
	Category and property type	

Table 1: Data overview

4.2 Data Cleaning and Preprocessing

Upon data collection, the project advanced to the data cleaning and preprocessing stage, which was implemented using Apache Spark for its scalability. This process included the integration of datasets from three separate games into three respective tables containing tailored information for each game.

The review data text was cleaned for clarity, and the data types for play hours and review length were corrected to ensure consistency. The information table merged all datasets into a single comprehensive structure. The information data was combined into one dataset and put into a comprehensive structure, with standardised column names and data types for in-game counts converted to integers for uniformity. The pricing data was also combined, and data types were verified to maintain accuracy.

Post-processing, the cleaned and structured data was saved in Parquet files. Acknowledging the importance of efficient data storage and retrieval, the Parquet file format was used to store the processed data. Parquet's columnar storage approach is space-efficient and significantly enhances query performance, making it an ideal choice for read-intensive operations. The benefits of Parquet include its suitability for storing big data of any kind and increased data throughput and performance (Databricks, 2024). By adopting Parquet, data compression and encoding schemes can be facilitated quickly and efficiently.

5. Database

5.1 Schema Design

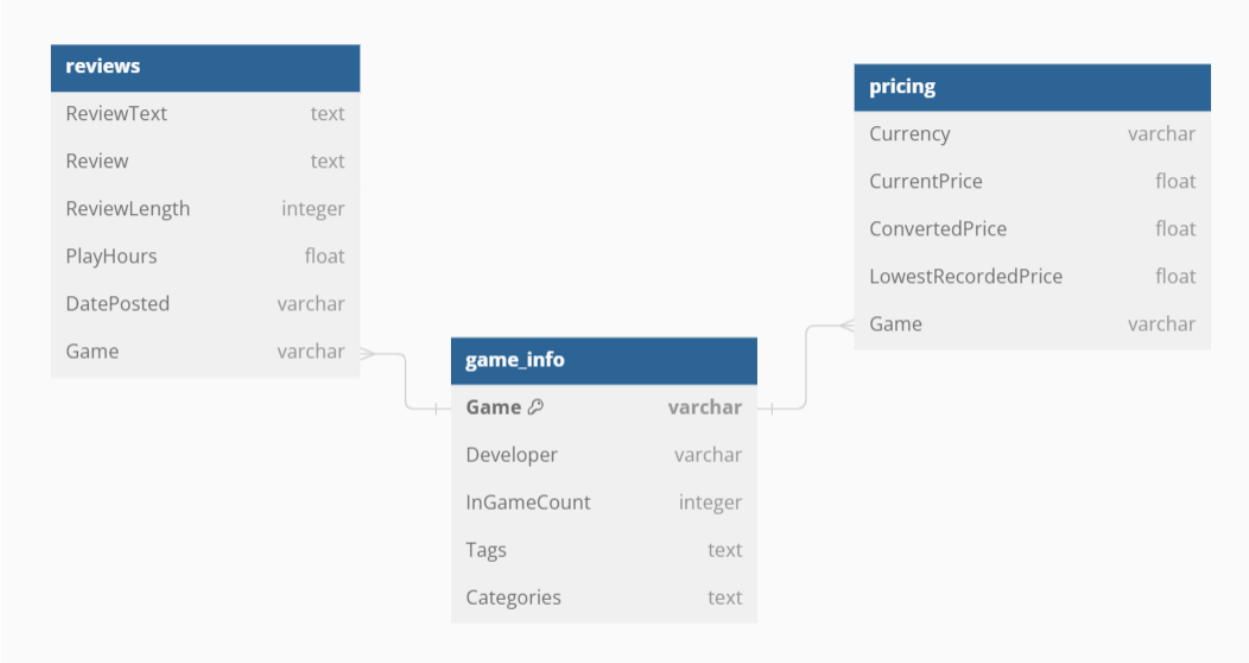

Figure 2: Schema structure

1. Table reviews:

   This table stores data about game reviews. It includes the full text of the review (ReviewText), a shorter version or summary (Review), the length of the review in words or characters (ReviewLength), the number of hours played (PlayHours), the date the review was posted (DatePosted), and a reference to the game being reviewed (Game), which links to the game_info table.

2. Table pricing:

   This table contains information about the pricing of games. It includes the type of currency used (Currency), the current price of the game (CurrentPrice), the price converted to a standard currency for comparison (ConvertedPrice), the lowest recorded price for the game (LowestRecordedPrice), and a reference to the corresponding game (Game), which links to the game_info table.

3. Table game_info:

   The game_info table serves as a repository for basic information about each game. It has a unique identifier for each game (Game), the developer of the game (Developer), the count of in-game items or features (InGameCount), the tags associated with the game which may describe the genre or characteristics (Tags), and the categories which could include the type of gameplay or content (Categories).

5.2 Queries

After successfully storing the database in Parquet format, which ensures optimised access speed and data efficiency, a set of SQL queries was developed to extract meaningful insights from the datasets. These queries enable comparative analysis with the result from retrieval-augmented generation (RAG) techniques and language learning models (LLM). The following queries were crafted to serve specific analytical purposes:

Find the Maximum and Minimum Prices for Each Game in Pounds:

This query is designed to retrieve each game's highest and lowest price points when priced in GBP. It provides an understanding of each game's pricing range and market positioning.

Average Play Hours per Game:

A query that calculates the mean playtime hours that players have spent on each game. This average can indicate a game's engagement level and overall player dedication.

Listing Developers and Their Games:

This query compiles a list of game developers alongside the titles they have developed, offering insights into the output and range of each developer's portfolio.

Count the Total of Review Text per Game:

A count of all the review texts for each game is tabulated in this query, providing a quantitative measure of the volume of feedback each game has received.

Counting Total 'Recommended' and 'Not Recommended':

This query tallies the number of 'Recommended' versus 'Not Recommended' tags within the review dataset for each game, giving a direct metric of the game's reception among players.

These queries form the baseline against which the nuanced text interpretations and answer generations of RAG and LLM can be evaluated, determining the practicality and efficiency of AI-enhanced data analysis in the context of game analytics.

6. Retrieval-Augmented Generation

6.1 RAG Implementation

For the Retrieval-Augmented Generation (RAG) component of the project, the AI8 platform is utilised, which provides free API access for this purpose. RAG combines the strength of information retrieval with the generative capabilities of language models to produce contextually informed responses.

The methodology adopted involves crafting a context from the data, formatted as a string, which acts as the knowledge source for the RAG process. Each query or question posed to the system is preceded by this context, allowing the model to generate responses based on the information provided. The operational syntax for this process is a string format that integrates the context with the question:

f"Based on this: '{context}', answer this: {question}, if the information is not from context, say 'I don't know'"

In this framework, the context is a string that represents the source data pulled from the Parquet files, effectively transformed into a textual format that the RAG system can process. The question is the input provided by the user.

The parquet files are then converted into string so the prompt can process it. However, when the review sources was used, instead of answer, the prompt will send an error saying

‘{'statusCode': 400, 'body': b'{"message": "Input too long!"}'}’

This signifies that the input string, including the context derived from the review sources, exceeded the maximum length the AI8 platform's API can handle at one time. To fix this issue, a splitting method was implemented to divide the lengthy source data into manageable segments.

The method involves partitioning the source text into several smaller chunks, each falling within the character or token limits imposed by the language learning model (LLM). Doing so allows each chunk to be individually processed by the RAG system without triggering input length errors. This approach requires careful segmentation so that each chunk remains coherent and contextually intact, allowing for the generation of meaningful and accurate responses.

The chunk is named df1 until df6, and here is the definition for each
#source 1 (df1) = review of lethal companies
#source 2 (df2) = review of palworld
#source 3 (df3) = review of craftopia
#source 4 (df4) = information about three games in general
#source 5 (df5) = pricing of three games
#source 6 (df6) = total of recommended and not recommended of each game

Moreover, additional logic is introduced to ensure that the chunks maintain the continuity of the information (df6). This method will overcome the input length limitation while leveraging the extensive data available in the review sources.

The next step is to make a model that can automatically pick the best source instead of manually choosing which source to use. In this case, the embedding method and cosine similarity were used to facilitate the process. Below are the details of the process.

- **Embedding the Question:**

The first step involves transforming the question into an embedding. An embedding is a numerical representation of text data that captures semantic meanings in a high-dimensional space. By converting the question into an embedding, the system can process it in a format amenable to comparison with other embedded texts.

- **Embedding the Sources:**

Similarly, the potential sources, which have been previously segmented into manageable chunks, are also converted into embeddings. This process ensures that each source text is represented as a vector in the same high-dimensional space as the question.

- **Computing Similarity:**

With both the question and the sources embedded, the next step is to compute the similarity between the question embedding and each source embedding. This is typically done using similarity metrics such as cosine similarity, which measures the cosine of the angle between two vectors. The idea is that the smaller the angle, the greater the similarity.

- **Selecting the Best Match:**

Based on the similarity scores, the source chunk with the highest similarity to the question embedding is identified as the best match. This chunk is presumed to contain the most relevant information to answer the question.

- **Querying the LLM:**

The selected source chunk is then used as the context in the RAG system, where the formatted question is presented to the language learning model (LLM) along with this context. The LLM generates an answer based on the information from the selected source, ensuring that the response is as accurate and informative as possible.

6.2 RAG Result

Task	Manual query	RAG and LLM
Find the maximum and minimum of prices for each game in pounds	Game |Max Price|Min Price| +----------+---------+---------+ | Craftopia | 14.76| 4.02| | Palworld | 26.5| 7.47| |Lethal Companies| 9.66| 3.29|	For Lethal Companies: Maximum Converted Price: 9.66 (Swiss Franc) Minimum Converted Price: 3.29 (Russian Ruble) For Palworld: Maximum Converted Price: 26.5 (Swiss Franc) Minimum Converted Price: 7.47 (South Asia - USD) For Craftopia: Maximum Converted Price: 14.76 (Swiss Franc) Minimum Converted Price: 4.02 (South Asia - USD)

Average Play Hours per Game	Game|Average Play Hours| +----------+----------+ | Craftopia|52.26| | Palworld| 72.92| |Lethal Companies| 42.10|	*provided a bunch of code and wrong answer
Listing Developers and Their Games	Developer |Games | +----------+----------+ |Zeekerss |[Lethal Companies] | |Pocketpair|[Palworld, Craftopia]|	"...games developed by Pocketpair listed in the provided data are Craftopia and Palworld. …or other developers mentioned, Zeekerss developed Lethal Companies"
Counting total 'Recommended', 'Not Recommended' and Count the total of review text per game	Game|Not Recommended|Recommended| +----------+----------+----------+ | Craftopia| 60| 90| | Palworld| 9| 141| |Lethal Companies| 8| 142|	For Craftopia: Average Review Length: 304.02666666666664 Total Recommend: 90 Total Not Recommend: 60 For Palworld: Average Review Length: 110.18666666666667 Total Recommend: 141 Total Not Recommend: 9 For Lethal Companies: Average Review Length: 84.10666666666667 Total Recommend: 142 Total Not Recommend: 8

Table 2: Result overview

The performance of Retrieval-Augmented Generation (RAG) and Language Learning Models (LLMs) in this project showcases both the strengths and limitations of these approaches compared to manually query databases.

Figure 3: RAG and LLM in extracting information from sources

Strengths in Extracting Information from Sources

- **Accuracy and Relevance:**

RAG and LLMs excel in extracting pertinent information from the provided sources. They are adept at navigating the embedded contexts to pull out relevant facts and details that directly address the queries posed. This capability stems from their design to efficiently understand and process natural language data.

- **Providing Additional Context:**

Beyond merely answering questions, these systems often enrich responses with additional context. This enhancement is particularly valuable as it offers a broader perspective or background, helping to deepen the user's understanding of the subject matter. This aspect highlights the generative capability of LLMs, which can interpolate and extrapolate information that might not be explicitly outlined in the query or the source data.

```
question = """
What is the average play hours craftopia?
"""
context = select_best_source(question, sources)

full = f"Based on this:'{context}', answer this: {question}, if the information is not from context, say 'I don't know'"

msg = get_resp_oai(full, "gpt-4-0125-preview")

display(Markdown("<div style='color: #34568B;'>\n\n" + msg))
```

```
Selected Source: df3
```

To find the average play hours for Craftopia from the provided data, we can sum up the total hours played and divide it by the number of reviews given. Let's calculate it accordingly:

- Sum of play hours: 0.9 + 63.7 + 35.0 + 22.7 + 3.3 + 79.9 + 10.8 + 57.9 + 14.0 + 1.3 + 69.0 + 14.6 + 32.4 + 0.3 + 124.4 + 135.4 + 1.0 + 8.1 + 10.2 + 7.4 + 59.0 + 0.6 + 16.6 + 52.7 + 58.1 + 8.5 + 69.2 + 50.0 + 12.1 + 24.4 + 73.7 + 17.0 + 21.5 + 30.4 + 22.2 + 6.8 + 27.5 + 43.5 + 20.3 + 5.5 + 20.3 + 55.5 + 3.4 + 23.7 + 233.4 + 62.8 + 62.0 + 221.4 + 87.6 + 0.6 + 73.8 + 0.3 + 49.0 + 3.7 + 7.0 + 53.8 + 17.9 + 77.1 + 14.9 + 35.1 + 5.6 + 183.3 + 7.4 + 16.9 + 74.2 + 1.0 + 205.0 + 21.1 + 8.6 + 73.4 + 1.1 + 57.8 + 0.3 + 9.5 + 11.0 + 8.0 + 235.0 + 1.2 + 20.8 + 108.9 + 224.2 + 17.3 + 44.1 + 45.0 + 45.7 + 19.5 + 7.1 + 112.3 + 75.7 + 65.3 + 13.6 + 6.6 + 27.2 + 8.5 + 31.8 = 4763.5 hours
- Number of entries: 157

Average play hours = Total play hours / Number of entries = 4763.5 / 157 = Approximately 30.34 hours

Therefore, the average play hours for Craftopia, based on the provided data, is approximately 30.34 hours.

Figure 3: RAG and LLM in calculating information

Limitations in Calculating Information

- **Inconsistent Calculations:**

RAG and LLMs sometimes get the result wrong when performing calculations or processing numerical data. They may attempt to generate calculations independently, but these can be inaccurate.

- **Erroneous Code Generation:**

Sometimes, these models provide code snippets intended to perform specific calculations. However, the accuracy of these generated codes is only sometimes correct. The generated code might contain logical or syntactic errors.

7. Closing

7.1 Conclusion

This project illustrates how the combined strengths of LLMs in generating coherent, context-aware responses and RAG's capability to anchor these responses in concrete and authoritative content can significantly enhance the reliability of generated information. The

"hallucination" limitation can be mitigated by leveraging a retrieval system that feeds relevant, vetted information into the generative process.

This project's methodical approach—spanning data collection, cleaning, and the strategic implementation of AI technologies—illustrates a proactive response to both the potential and the pitfalls of contemporary AI applications. Focusing on enhancing data integrity and leveraging the comprehensive data environment of Steam sets a precedent for future AI-driven projects in terms of scalability, ethical compliance, and precision.

Based on the comparative result from the manual query and gathering information from LLM and RAG, it seems the strength of this approach is in extracting information from sources, as the AI provides high accuracy and relevance while providing additional context, which manual query does not. On the other hand, the limitations are in calculating and generating new information, as the AI has inconsistent calculations and sometimes provides incorrect code.

7.2 Limitations and Further Recommendations

Limitations

1. **API Constraints:** The current project utilises a free API from the AI8 platform, which limits the amount of data that can be processed and the computational resources available. This restriction may affect the performance and scalability of the models used in the project.
2. **Data Volume:** The dataset used in this project is limited in scope. While sufficient for initial exploration and analysis, a more extensive dataset could provide more insights and enhance the model's accuracy and generalizability.
3. **Data Chunking Inequality:** The method used to split data into chunks for processing by the RAG system is not uniform, which can lead to inconsistencies in source relevance. Some chunks may be weighted more heavily than others, skewing the model's ability to evaluate and utilise all available information fairly.

Further Recommendations

1. **Upgrading API Access:** To overcome the limitations imposed by the free API, future projects should consider investing in a paid subscription with the AI8 platform or exploring other more robust API services. This change would likely provide higher processing capabilities and access to more advanced features, enhancing the overall effectiveness of the project.
2. **Expanding the Dataset:** Increasing the volume and variety of data could significantly improve the project's outcomes. More data would not only help train the models more comprehensively but also aid in achieving better generalisation across different scenarios. This expansion could involve incorporating additional metrics from Steam.
3. **Improving Chunk Equitability:** To address the issue of unequal data chunking, it is recommended that more data be provided so the 'general information' chunk can have more context. This approach should ensure that each chunk is of comparable size and content density, allowing for a more balanced retrieval process. Implementing algorithms that automatically adjust chunk sizes based on content complexity and relevance could be a valuable advancement.

These improvements will enhance the project's capacity to serve as a robust tool for effectively analysing gaming data and consumer behaviour.

7.3 Data Governance

It is crucial to highlight that the web scraping techniques employed in this project are utilised solely for educational purposes. This approach underscores a commitment to ethical data practices, ensuring that no collected data is used for commercial benefits. The primary objective is to create a learning environment conducive to exploring real-world datasets, thus cultivating analytical skills vital in Business Analytics.

The project incorporates several measures to maintain data integrity and privacy. This includes the anonymization of personal data elements such as user IDs and the careful handling of any sensitive information to prevent any misuse. Additionally, all API interactions are censored to safeguard operational details and user data from unauthorised access.

The infrastructure of this project is deliberately designed to be scalable and ethically compliant, allowing for seamless adaptation to more sophisticated data collection and analysis methodologies as they become necessary.

Citation

Baxter, K., & Schlesinger, Y. (2023). Managing the risks of generative AI. Harvard Business Review. From https://hbr.org/2023/06/managing-the-risks-of-generative-ai. Retrieved April 15, 2024.

Databricks (2024) What is Apache Parquet?, Databricks. Available at: https://www.databricks.com/glossary/what-is-parquet (Accessed: 13 March 2024).

Udofia, E. (2024, January 4). WebScrapping: BeautifulSoup or Selenium? Medium. From https://medium.com/@udofiaetietop/webscrapping-beautifulsoup-or-selenium-3467edb3c0d9. Retrieved April 15, 2024.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... & Stoica, I. (2016). Apache Spark: A unified engine for big data processing. Communications of the ACM, 59(11), 56-65. doi:10.1145/2934664. Retrieved April 15, 2024.

ZenData. (2024, January 17). Enhancing LLM output with retrieval-augmented generation. From https://www.zendata.dev/post/enhancing-llm-output-with-retrieval-augmented-generation . Retrieved April 15, 2024.