Big Data Project Report: Steam Reviews Analysis Dashboard

BIG DATA (CSGY–6513)

Rohan Gore (rmg9725)
Pooja Thakur (put2003)

May 15, 2025

Project Repository:
https://github.com/rohan-g0re/bigdata_project/tree/main/

Contents

1 INTRODUCTION

1.1 General Introduction

Steam, the largest digital game distribution platform, hosts over 100 million user reviews across thousands of games. These reviews are rich in metadata, including playtime, language, purchase type, and upvotes. This project focuses on processing, analyzing, and visualizing a large dataset of Steam game reviews. The primary goal is to extract meaningful insights from the review data, such as review trends over time, sentiment analysis, game acquisition methods, and top-performing games. These insights are presented through an interactive and multi-tab dashboard, allowing Business Analysts and Game Developers to explore the data dynamically. The dataset structure includes fields related to game reviews, authors, timestamps, and voting behavior.

1.2 Motivation

With millions of reviews spread across thousands of games, manually browsing Steam reviews is not only inefficient but often overwhelming. Users seeking honest feedback or developers monitoring public reception must wade through unstructured, multilingual, and often noisy content. This project addresses that challenge by providing a scalable solution to process and summarize this vast sea of data. From an analyst's perspective, the dashboard allows individuals to explore which games have consistently received attention and positive feedback across different time periods, regions, and purchase conditions (e.g., free vs. paid). For game developers, it acts as a powerful feedback tool, enabling them to gauge community sentiment, identify spikes in engagement during sales or updates, and better understand how player experiences translate into reviews. In doing so, the project bridges the gap between raw feedback and actionable insights through automation, analytics, and intuitive visualization.

1.3 Objective

The goal of this project is to make sense of the massive sea of reviews on Steam by turning them into clear, useful insights. The dashboard is built around two main features:

- The **Engagement Tab** helps users explore which games have the most reviews, with options to filter by year and language. It's a quick way to see what's been trending and when.
- The **Game Info Tab** goes deeper. You can search for any game and get a full breakdown — total reviews, how much time players spent before writing them, how many were written during Early Access, and even the overall sentiment of the reviews. We also run sentiment analysis using an AI model to highlight the most positive and most negative reviews.

All of this is visualized through clean graphs and charts so users don't have to dig through raw data. Everything runs on tools that can handle big data, so it stays fast even with millions of records. Whether you're a gamer curious about a title or a developer looking for feedback, this dashboard makes exploring reviews easier and more insightful.

2 METHODOLOGY

2.1 Architecture and Workflow

1. The project's architecture is built to handle large-scale review data efficiently from end to end. It begins with data ingestion, where raw Steam review data, stored in CSV format is converted to Apache Parquet format with Snappy Compression. It is then loaded into a Spark 3.4.0 environment.

2. The preprocessing stage leverages PySpark 3.4.0, utilizing its DataFrame API along with an analysis derived python script, for distributed transformations to clean and prepare the dataset. This includes dropping unnecessary columns, casting string-based fields to appropriate data types like integers or booleans, converting Unix timestamps, and handling null values. After cleaning, the refined data is written back to disk as a new directory of Parquet files, compressed using Snappy (Spark's default) and partitioned for optimized downstream use.

3. The frontend application is built using Streamlit 1.28.0 and organized into two primary tabs: the Engagement Tab and the Game Info Tab. These tabs are rendered by `streamlit_app.py`, which also initializes a shared SparkSession for backend computations via a utility function in `src/utils/spark_session.py`. This session is configured to run in `local[4]` mode (utilizing 4 cores on the local machine) with a default driver memory of 8GB (`spark.driver.memory`), ensuring efficient local processing. In the Engagement Tab, the app groups the dataset by game, counts the number of reviews for each, and sorts them to identify the most discussed titles. The Game Info Tab takes user input in the form of a game name and a time window, filters the data accordingly, and computes useful statistics like total review count, average playtime at review time, and early access review breakdown.

4. A core part of the Game Info Tab is sentiment analysis. The system loads a pre-trained DistilBERT model, specifically `distilbert-base-uncased-finetuned-sst-2-english` from the Hugging Face Transformers 4.35.0 library, chosen for its balance of performance and accuracy on sentiment classification tasks. Review text is tokenized with a maximum sequence length of 128 tokens, with truncation and padding applied as necessary. Up to 1000 reviews are sampled per query (prioritizing English due to the model choice and then selecting the most upvoted), then processed in batches of 32. Each review is scored between -1 and +1, with scores above 0.2 labeled as positive and below -0.2 as negative. These results are used to highlight the most upvoted reviews from each sentiment group and to calculate an overall confidence-weighted sentiment score for the game.

5. All insights are presented through interactive visualizations built using Altair and Plotly — including treemap for top games, time-series plots for daily review activity, pie charts showing how users acquired the game (free vs. paid), and both gauge and histogram charts to depict sentiment distribution. Users interact with the system through Streamlit's widgets like sliders, buttons, text inputs, and language filters. These features collectively create an intuitive, real-time analytical experience that makes exploring Steam reviews both insightful and engaging.

2.2 Workflow Diagram

2.2.1 System Architecture Overview:

2.2.2 Detailed Workflow:

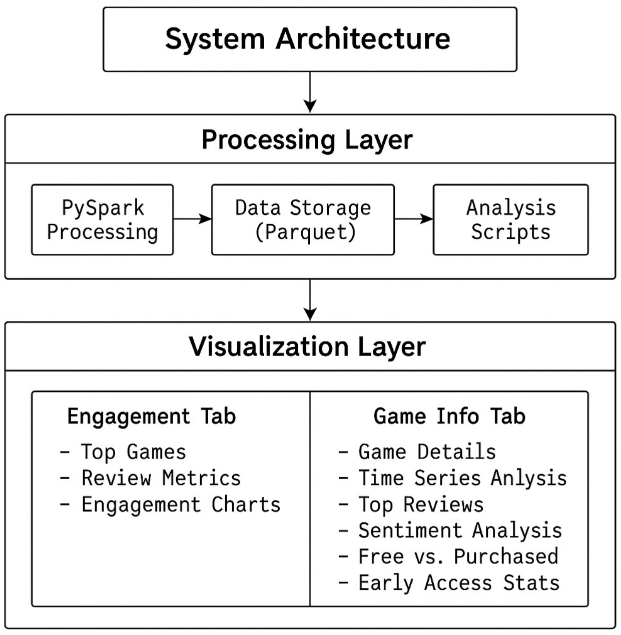
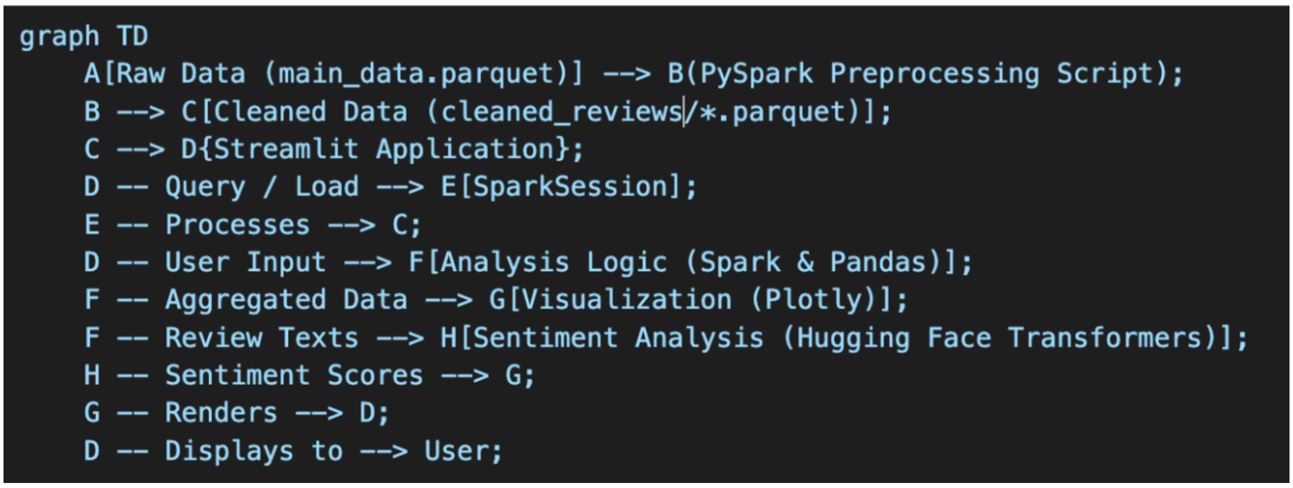


Figure 1: System Architecture Overview



```
graph TD
    A[Raw Data (main_data.parquet)] --> B(PySpark Preprocessing Script);
    B --> C[Cleaned Data (cleaned_reviews/*.parquet)];
    C --> D{Streamlit Application};
    D -- Query / Load --> E[SparkSession];
    E -- Processes --> C;
    D -- User Input --> F[Analysis Logic (Spark & Pandas)];
    F -- Aggregated Data --> G[Visualization (Plotly)];
    F -- Review Texts --> H[Sentiment Analysis (Hugging Face Transformers)];
    H -- Sentiment Scores --> G;
    G -- Renders --> D;
    D -- Displays to --> User;
```

Figure 2: Detailed Workflow Diagram

2.3 Data Preprocessing Pipeline

The preprocessing pipeline begins by loading raw review data into a Spark environment. A SparkSession is initialized, configured for local execution (`local[4]`) with an 8GB driver memory allocation (`spark.driver.memory`), as defined in `src/utils/spark_session.py`, to ensure efficient handling of the large dataset. The data, stored in `main_data.parquet`, initially contains a wide variety of fields, many of which are either redundant or poorly typed. For example, several numeric and boolean values are stored as strings, and timestamp fields appear as Unix epoch values. This initial structure necessitates careful schema correction before any meaningful analysis can be performed.

2.3.1 Initial Data Loading & Schema

Raw Steam review data is ingested from Apache Parquet files (e.g., `main_data.parquet`). The initial schema often includes a mix of string, numeric, and boolean data, where numeric and boolean values are frequently stored as strings, and timestamps are in Unix epoch format.

2.3.2 Column Pruning

To clean and streamline the dataset, a column pruning step removes fields that add little analytical value or create redundancy. This includes `author_playtime_last_two_weeks`, `recommendationid`, `appid`, `voted_up`, `hidden_in_steam_china`, and `steam_china_location`.

2.3.3 Data Type Conversion

Once the schema is trimmed, the script proceeds with explicit data type conversions using PySpark's `cast()` method. Playtime, vote counts, and other numeric metrics are cast from string to IntegerType. Flags such as `steam_purchase`, `received_for_free`, and `written_during_early_access` are cast to BooleanType. Unix timestamp fields like `author_last_played`, `timestamp_created`, and `timestamp_updated` are converted into proper TimestampType fields using PySpark SQL functions like `to_timestamp(from_unixtime(col(...).cast("long")))`.

2.3.4 Null Value Handling

The next step is rigorous null value handling. The script checks for missing (NULL), NaN (for numeric types), or empty-string values (for string types) across key analytical columns (e.g., 'review', 'game', 'timestamp_created'). Any row containing such incomplete data is removed using PySpark's `dropna()` function on a subset of critical columns, or a more comprehensive filter condition to preserve the integrity of downstream analytics.

2.3.5 Final Processed Data Output

After this cleansing step, the final dataset schema is printed to confirm the structure is clean, consistent, and ready for analysis. The processed data is saved in a new location as a set of Parquet files using Snappy compression (Spark's default) in `overwrite` mode. This final cleaned dataset becomes the input for all dashboard functionalities.

2.3.6 Final Data Schema

The resulting schema after preprocessing is:

```
|-- game: string
|-- author_steamid: string
|-- author_num_games_owned: integer
|-- author_num_reviews: integer
|-- author_playtime_forever: integer
|-- author_playtime_at_review: integer
|-- author_last_played: timestamp
|-- language: string
|-- review: string
|-- timestamp_created: timestamp
|-- timestamp_updated: timestamp
|-- votes_up: integer
|-- votes_funny: integer
|-- weighted_vote_score: float
|-- comment_count: integer
|-- steam_purchase: boolean
|-- received_for_free: boolean
|-- written_during_early_access: boolean
```

2.4 Tools Used – Summary Table

Layer	**Tool**	**Purpose**
Storage	Parquet	Optimized large-scale storage
Processing	PySpark	Distributed data transformations
NLP	DistilBERT (SST-2), NLTK	Sentiment classification & stop-word filtering
Visualization	Streamlit, Plotly, Altair	UI and interactive graphs
Backend	Python (via Streamlit app script)	Orchestrate business logic

Table 1: Tools Used in the Project

2.5 Feature Implementation

The dashboard is built around two main interactive tabs — Engagement and Game Info — each offering a different angle of exploration into the Steam reviews dataset.

The **Engagement Tab** is designed to give users a high-level view of the most reviewed games on Steam. Users can choose how many games they want to view using a slider input (ranging from 5 to 50, with a default value of 10). Once selected, the dashboard displays a ranked table of the top N games based on the number of reviews. To make the data more engaging, a bar chart (or treemap as mentioned in 2.1) is also generated to visually represent the review counts for each game. Additionally, the tab highlights basic stats, such as which game received the most reviews and the average review count across the top games shown. We also introduced a leaderboard showcasing the top N most influential players, ranked using a Steam-generated influence score based on their total number of reviews and the upvotes those reviews received.

The **Game Info Tab** dives deeper into the specifics of a single game. Users can type in the name of a game and apply filters such as a date range (start and end dates) and a language dropdown to narrow down the reviews they're interested in. Once the data is filtered, several key metrics are displayed — including the total number of reviews, the average playtime at the time of review, the number and percentage of Early Access reviews, and an overall sentiment rating, which is labeled descriptively (e.g., "Very Positive", "Mixed", etc.) based on the underlying sentiment score.

This tab also includes a variety of visualizations to bring the data to life. A sentiment gauge shows the game's overall sentiment score on a scale from very negative to very positive, while a histogram reveals the distribution of sentiment scores across reviews. A pie chart illustrates the proportion of users who purchased the game versus those who received it for free. There's also a time series chart that tracks daily review counts, helping users spot trends like review spikes during updates or sales.

Lastly, the Game Info Tab features a rich section of review cards. These cards highlight the most upvoted, most commented, and most funny-tagged reviews. To complement this, the dashboard uses AI-driven sentiment analysis to pull up the most positive and most negative reviews, letting users see real examples of praise or criticism. Each card is styled to reflect the review's tone, playtime, and engagement level, making it easy to understand what real players are saying.

3 RESULT

One of the key analytical components of this project is the sentiment scoring of user reviews. Using the DistilBERT SST-2 model, each review is analyzed and assigned a sentiment score ranging from -1 (strongly negative) to +1 (strongly positive). Reviews are categorized as positive if their score is ≥ 0.2, negative if ≤ -0.2, and neutral if they fall in between. This approach gives us a scalable and meaningful way to understand the emotional tone of thousands of reviews without manual labeling.

Through these analyses, several game-related insights emerged. Notably, games that were offered for free tended to attract more critical feedback — possibly due to lower expectations or broader reach among users unfamiliar with the genre. In contrast, paid titles often showed more balanced sentiment, especially among players with longer playtimes. Additionally, the data revealed clear spikes in review activity around major sales events, game updates, or community controversies, all visible in the time-series review graphs.

From a performance perspective, the system is designed to remain responsive even while handling large volumes of data. The sentiment analysis pipeline processes up to 1000 reviews per query to strike a balance between depth and speed, and this batch is sampled to prioritize English reviews for better accuracy. Meanwhile, PySpark handles data filtering and aggregations efficiently — most common queries (e.g., top games, review stats per game) are completed in under a minute. Together, these optimizations ensure a seamless user experience while preserving analytical depth.

4 CODE

1. The codebase is modular and well-structured, organized into logical components that make it easy to navigate and extend. At the core is `streamlit_app.py` (approx. 121 lines), which acts as the entry point for the entire application. This script initializes the Streamlit UI (using `st.set_page_config`, `st.title`, `st.tabs`), applies custom CSS, and orchestrates the rendering of the ”Engagement” (previously referred to as ”Top Games”) and ”Game Info” tabs, providing users with distinct entry points to explore the data.

2. The **Engagement Tab** is primarily driven by logic within `src/tabs/engagement_tab/ui.py` (assuming the name change from ”top_games_tab”) and its associated analysis script. This module contains the logic for rendering the UI components and visualizations that display the top-reviewed games. It handles user input (like the number of top games to show) and prepares data for plotting both the review table and corresponding chart (e.g., treemap or bar chart) using Plotly or Altair.

3. For more detailed analysis, the **Game Info Tab** is built using `src/tabs/game_tab/ui.py`, which manages all user interactions including game name input, date filters, and language selection. It also coordinates the display of review statistics, sentiment scores, and multiple visualizations. Supporting this tab is the logic-heavy `src/tabs/game_tab/analysis.py` (approx. 230 lines), which connects to Spark to filter and aggregate the review data based on user input. It uses PySpark DataFrame operations such as `filter()` (for game name and date ranges), `groupBy().agg()` (for statistics like `avg_playtime`, review counts per day), `withColumn()` (for adding `review_date`), and `orderBy().limit()` (for fetching top reviews). It then typically collects results into Pandas DataFrames for easier integration with Streamlit and Plotly/Altair.

4. Meanwhile, `src/tabs/game_tab/sentiment_analysis.py` (approx. 120 lines) is responsible for loading the sentiment model and performing batch sentiment classification. The `load_sentiment_model` function uses Streamlit’s `@st.cache_resource` to load and cache the `distilbert-base-uncased-finetuned` model and its tokenizer from Hugging Face. The `analyze_sentiment_batch` function processes review texts in batches (default size 32), performing tokenization (max length 128, truncation, padding) and model inference using `torch.no_grad()` for efficiency during inference. It calculates sentiment scores (positive probability - negative probability) and confidence. The `get_overall_sentiment` function then computes a confidence-weighted average sentiment for

the game, and `get_extreme_reviews` identifies the most upvoted positive and negative reviews based on these scores.

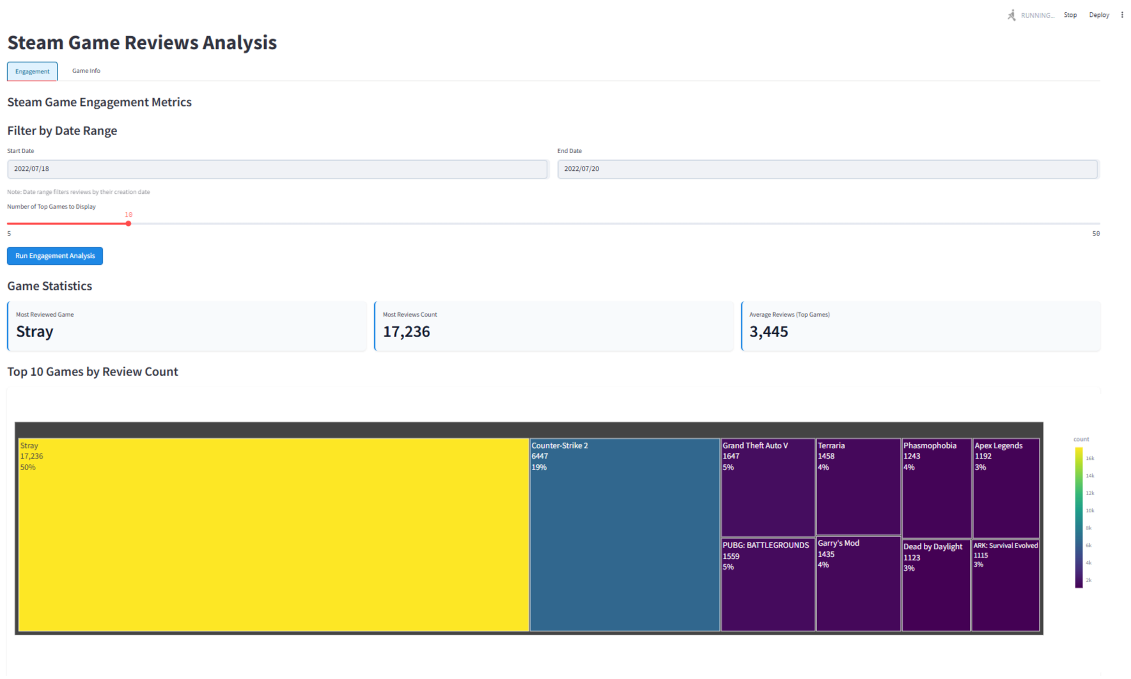
5. Visualizations, including Plotly gauge charts, time series plots, and pie charts, (and Altair treemaps if applicable) are defined and generated within the respective UI or analysis scripts (e.g., within `src/tabs/game_tab/ui.py` or by functions in `src/tabs/game_tab/analysis.py` preparing data for these libraries). These charts are designed to be visually informative while keeping the dashboard responsive and clean. Finally, `src/utils/spark_session.py` (approx. 20 lines) provides a centralized `get_spark_session` utility to initialize and configure a SparkSession (appName `SteamReviewsAnalysis`, master `local[4]`, driver memory default '8g'), ensuring consistent access to Spark capabilities across all modules.

5 SCREENSHOTS

This section displays various screenshots from the implemented dashboard, showcasing its key features and user interface.

5.1 Engagement Tab Screenshots

The Engagement Tab provides a high-level overview of game popularity and user interaction.



Steam Game Reviews Analysis

Steam Game Engagement Metrics

Filter by Date Range

Start Date: 2022/07/18

End Date: 2022/07/20

Note: Date range filters reviews by their creation date

Number of Top Games to Display: 10 (5–50)

Game Statistics

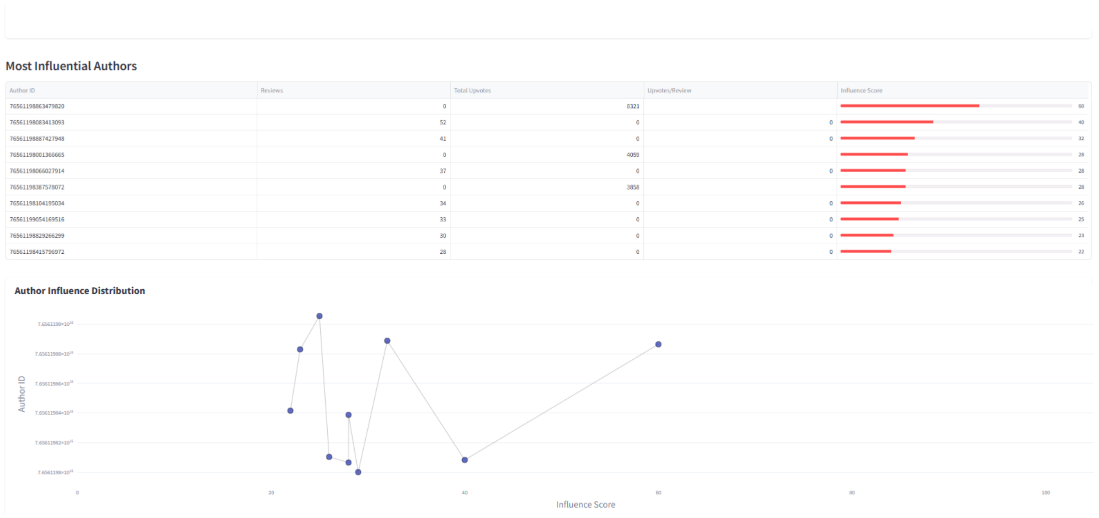
Most Reviewed Game: Stray

Most Reviews Count: 17,236

Average Reviews (Top Games): 3,445

Top 10 Games by Review Count

Figure 3: Engagement Tab: Main view displaying top reviewed games.



Most Influential Authors

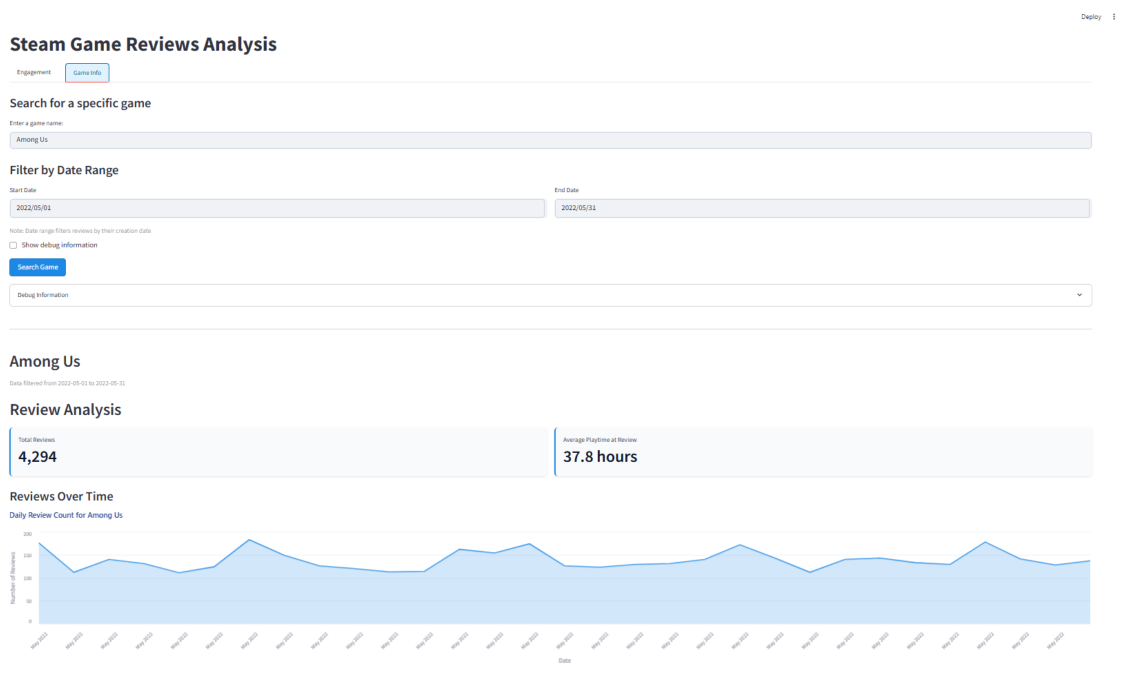
Author ID	Reviews	Total Upvotes	Upvotes/Review	Influence Score
76561198863479820	0	8321		60
76561198083413093	52	0	0	40
76561198887427948	41	0	0	32
76561198001366665	0	4059		28
76561198066027914	37	0	0	28
76561198387578072	0	3858		28
76561198104195034	34	0	0	26
76561199054169516	33	0	0	25
76561198829266299	30	0	0	23
76561198415796972	28	0	0	22

Author Influence Distribution

Figure 4: Engagement Tab: Influential Author Analysis and Influence Distribution Graph.

5.2 Game Info Tab Screenshots

The Game Info Tab allows for a deep dive into individual game review data, including sentiment analysis and specific review examples.



Steam Game Reviews Analysis

Search for a specific game

Enter a game name: Among Us

Filter by Date Range

Start Date: 2022/05/01

End Date: 2022/05/31

Note: Date range filters reviews by their creation date

Among Us

Data filtered from 2022-05-01 to 2022-05-31

Review Analysis

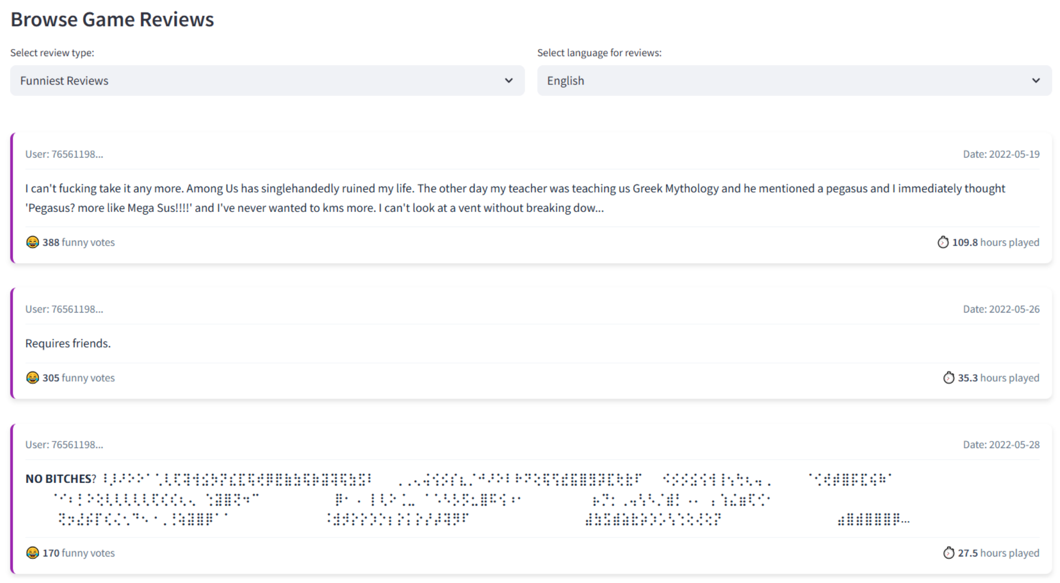
Total Reviews: 4,294

Average Playtime at Review: 37.8 hours

Reviews Over Time

Daily Review Count for Among Us

Figure 5: Game Info Tab: Overview display for a selected game, showing total reviews and average playtime in selected time frame.



Browse Game Reviews

Select review type: Funniest Reviews

Select language for reviews: English

User: 76561198... Date: 2022-05-19

I can't fucking take it any more. Among Us has singlehandedly ruined my life. The other day my teacher was teaching us Greek Mythology and he mentioned a pegasus and I immediately thought 'Pegasus? more like Mega Sus!!!!' and I've never wanted to kms more. I can't look at a vent without breaking dow...

388 funny votes 109.8 hours played

User: 76561198... Date: 2022-05-26

Requires friends.

305 funny votes 35.3 hours played

User: 76561198... Date: 2022-05-28

NO BITCHES?

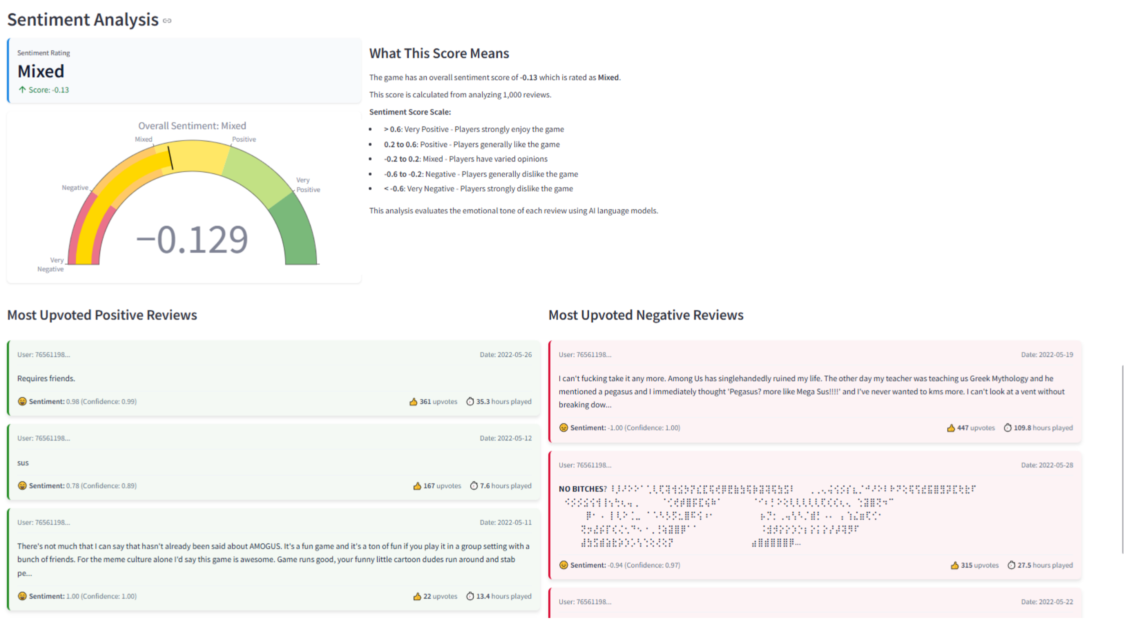
170 funny votes 27.5 hours played

Figure 6: Game Info Tab: Filterable reviews by type and language.

6 CONCLUSION & FUTURE SCOPE

6.1 Conclusion

This project presents a complete pipeline for extracting insights from large-scale Steam review data — from raw Parquet ingestion to an interactive dashboard backed by PySpark and NLP. The combination of distributed data processing, transformer-based sentiment analysis, and a clean Streamlit interface allows users to explore games in rich, customizable ways. Whether it's understanding how a game's perception evolved over time or surfacing the most impactful user reviews, the dashboard bridges raw user feedback with actionable patterns. For developers, gamers, or data scientists, it provides a powerful lens into the voice of the gaming community.


Sentiment Analysis

Sentiment Rating
Mixed
↑ Score: -0.13

What This Score Means

The game has an overall sentiment score of **-0.13** which is rated as **Mixed**.

This score is calculated from analyzing 1,000 reviews.

Sentiment Score Scale:

- **> 0.6**: Very Positive - Players strongly enjoy the game
- **0.2 to 0.6**: Positive - Players generally like the game
- **-0.2 to 0.2**: Mixed - Players have varied opinions
- **-0.6 to -0.2**: Negative - Players generally dislike the game
- **< -0.6**: Very Negative - Players strongly dislike the game

This analysis evaluates the emotional tone of each review using AI language models.

Most Upvoted Positive Reviews

User: 76561198... Date: 2022-05-26
Requires friends.
Sentiment: 0.98 (Confidence: 0.99) **361** upvotes **35.3** hours played

User: 76561198... Date: 2022-05-12
sus
Sentiment: 0.78 (Confidence: 0.89) **167** upvotes **7.6** hours played

User: 76561198... Date: 2022-05-11
There's not much that I can say that hasn't already been said about AMOGUS. It's a fun game and it's a ton of fun if you play it in a group setting with a bunch of friends. For the meme culture alone I'd say this game is awesome. Game runs good, your funny little cartoon dudes run around and stab pe...
Sentiment: 1.00 (Confidence: 1.00) **22** upvotes **13.4** hours played

Most Upvoted Negative Reviews

User: 76561198... Date: 2022-05-19
I can't fucking take it any more. Among Us has singlehandedly ruined my life. The other day my teacher was teaching us Greek Mythology and he mentioned a pegasus and I immediately thought 'Pegasus? more like Mega Sus!!!!' and I've never wanted to kms more. I can't look at a vent without breaking dow...
Sentiment: -1.00 (Confidence: 1.00) **447** upvotes **109.8** hours played

User: 76561198... Date: 2022-05-28
NO BITCHES? [illegible]...
Sentiment: -0.94 (Confidence: 0.97) **315** upvotes **27.5** hours played

User: 76561198... Date: 2022-05-22

Figure 7: Game Info Tab: Showcasing the AI-driven sentiment analysis with Sentiment Distribution Gauge and examples of most positive and negative review cards.

6.2 Future Scope

Looking ahead, there are several exciting directions to expand the tool's capabilities. One priority is to introduce multilingual sentiment analysis using models like XLM-RoBERTa, enabling deeper insights across global audiences. From a system perspective, scalability can be improved by integrating distributed task queues and containerized deployment via Docker with CI/CD pipelines. On the analytics front, features like topic modeling could help identify recurring themes in reviews, while time-series sentiment tracking could surface trends tied to updates or community events. Support for streaming reviews using Kafka and Spark Structured Streaming would enable real-time monitoring. Lastly, adding user accounts and personalized dashboards would allow bookmarking, history tracking, and custom filters — taking the dashboard from a tool to a full-fledged product experience.

7 REFERENCES

[1] Kaggle Steam Review Dataset (Specify exact dataset if possible, e.g., "Steam Reviews" by Andrew Shepard). https://www.kaggle.com/datasets/kieranpoc/steam-reviews (Example URL)

[2] Hugging Face Transformers Library. https://huggingface.co/transformers/

[3] DistilBERT SST-2 Model (fine-tuned). https://huggingface.co/distilbert-base-uncased-finetuned-

[4] Apache PySpark Documentation. https://spark.apache.org/docs/latest/api/python/

[5] Streamlit Documentation. https://docs.streamlit.io/

[6] Plotly Python Documentation. https://plotly.com/python/

[7] Altair Documentation. https://altair-viz.github.io/