Video Game Communities Knowledge Graph Analysis

Abstract

This project proposes solutions to the challenges faced by video game publishers in accessing and utilizing social media and other public user data for influencer marketing campaigns optimization. The proposed solutions include: 1) constructing a knowledge graph that consolidates data from multiple sources and links different actors in the gaming ecosystem. 2) applying graph analytics and social network analysis to identify key players within the gaming communities who can effectively spread the message to the target market. The project provides video game publishers with accessible, data-driven tools to gain valuable insights that help shape their content strategy, media positioning, and partnership evaluations.

Introduction and Question Formulation

Challenge

Game reviews, generated by various communities and influencers are important signals of game quality and have a significant impact on video game sales. In response, video game publishers are using various advertising tactics to target these gaming communities with influencer marketing in video games worth $4.4 billion in 2022. However, ineffective resource allocation and suboptimal decision-making are still prevalent in video game marketing due to several factors. Firstly, data about gaming communities is fragmented and spread out across various sources, which makes it challenging to access and analyze. Secondly, the current approach to user targeting disregards the primary appeal of modern video gaming, which is the ability to form communities around games. Consequently, the state of the art method for selecting influencers for partnerships is based on user properties like demographics and interests, with no consideration given to their network relationships. This can lead to missed opportunities to engage with potential customers. To address these challenges, the industry needs to adopt a new approach to data collection and analysis, and a more comprehensive targeting method for users and influencers.

Data science problem

The first data science challenge in this project entails integrating various data streams from multiple public APIs to build a comprehensive Knowledge Graph. Our ultimate goal is to build a recommendation algorithm to help video game publishers find influencers to sponsor so that their marketing ROI is maximized. Our central hypothesis is that this problem can be solved as a “Key

Player Problem" in the social network of influencers. The first step involves gathering data from these APIs and organizing it into a structured graph representation. This integration process requires careful consideration of data formats, APIs, and efficient data extraction techniques. To enhance the analysis, the raw data has to be enriched with new features that provide additional context and facilitate more comprehensive insights. Once the Knowledge Graph is constructed, the next challenge is to select and apply appropriate methods from graph analytics and network analysis to extract meaningful insights from the data. This involves utilizing techniques such as centrality measures, community detection, and other graph-based algorithms to uncover patterns, relationships, and key information within the Knowledge Graph. The goal is to gain a deeper understanding of the interconnectedness and dynamics present in the data. Finally, in order to effectively communicate these insights to end-users, the Knowledge Graph needs to be seamlessly integrated into a user-friendly graphical user interface (GUI) dashboard. This dashboard will serve as a platform for delivering analytics and visualizations to users, presenting the valuable insights derived from the Knowledge Graph in an accessible and intuitive manner.

Questions

Some of the questions that have been formulated in order to address our challenges are:

- How can data be efficiently extracted from multiple APIs on an ongoing basis to overcome API rate limits, quotas, and optimize for speed and cost?
- What methods can be employed to match entities from various data sources in order to construct a comprehensive knowledge graph?
- Which new features can be derived from the available raw data to provide valuable insights and enhance analysis?
- Which centrality measures should be considered to identify the most influential gamers within gaming networks?
- How can multiple centrality measures be combined to gain a more comprehensive understanding of influence dynamics and accurately identify top influencers?

Related work

Twitterrank: Finding topic-sensitive influential Twitterers
https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=1503&context=sis_research

Network Centrality: An Introduction
https://arxiv.org/pdf/1901.07901.pdf

Finding Influential Users in Social Media Using Association Rule Learning
https://arxiv.org/pdf/1604.08075.pdf

The Key Player Problem

http://www.casos.cs.cmu.edu/events/summer_institute/2006/reading_list/borgatti/Borgatti_Key_Player.pdf

Combined Centrality Measures for an Improved Characterization of Influence Spread in Social Networks
https://arxiv.org/pdf/2003.05254.pdf

Influence and Passivity in Social Media
https://www.hpl.hp.com/research/scl/papers/influence/influence.pdf

Machine Learning Techniques for brand-influencer Matchmaking on the Instagram Social Network
https://arxiv.org/pdf/1901.05949.pdf

Teams, roles and responsibilities

In the first phase of the project, Polina and Robert successfully completed the tasks related to problem identification, data extraction, and feature engineering. Their responsibilities were divided as follows:

Polina, as the project and data science manager, took charge of outlining the project roadmap, defining its scope, and managing the assigned tasks. As the solution architect, she designed the project's information architecture to facilitate effective consumption of insights. Polina also collected the assigned portion of required datasets and ensured the timely delivery of project milestones.

Robert focused on programming and data processing aspects of the project. He successfully extracted necessary data, processed data and implemented machine learning algorithms to generate new features. Robert also managed the project resources on AWS servers, including the database and all necessary tools. He contributed as the data analyst, and methods expert.

Moving into the second phase of the project, Polina took on the responsibilities of constructing the knowledge graph, conducting graph and network analysis, and building a user interface. In this phase, she played multiple roles to ensure the successful completion of these tasks.

Data Acquisition

Data Sources

Steam API

Steam is a popular gaming platform and social networking site that offers users the ability to purchase and store games. With approximately 1.7 billion PC gamers worldwide, Steam is responsible for 50% to 70% of all PC game downloads globally. The Steam API provides access to

various endpoints that allow developers to retrieve information about games, including their descriptions, release dates, developers, publishers, genres, tags, and Metacritic scores. Steam is one of the largest digital distribution platforms for PC gaming, with millions of active users worldwide. To accommodate the high velocity of user interactions, real-time updates, market activities, and event-driven traffic, the Steam API infrastructure is designed to handle the substantial data flow and respond efficiently to user requests. Steam returns data in structured format (JSON). However, the content of the reviews themselves contain unstructured text. The actual text of the reviews is user-generated and can vary greatly in terms of writing style, grammar, and formatting.

Twitch API

Twitch is a fast-growing streaming platform owned by Amazon that focuses on video games. The Twitch API allows developers to access information about users, streams, games, and other relevant data. Unlike other social media APIs, Twitch shares a lot of data publicly, including broadcaster's followers, chatters, and moderators. On Twitch, millions of users can stream their gameplay or watch streams from others in real-time. Consequently, the data velocity of the Twitch API is relatively high due to the dynamic nature of the platform and the constant flow of data being generated. Twitch API returns data in a structured format (JSON) with well-defined endpoints and clear organization. However, when it comes to Twitch stream titles extracted via the Twitch API, they can be considered unstructured data. Stream titles are typically free-form text entered by streamers to provide a brief description or context for their live streams. They often contain a mix of words, phrases, emojis, hashtags, and other text elements that streamers choose to use. These titles can vary significantly in length, style, and content, depending on the streamer's preferences.

Steamspy API

SteamSpy is a third-party service that provides estimates and statistics about games available on the Steam platform. For this project goals, Steamspy API supplements Steam data by providing additional insights about games like score rank of the game based on user reviews, average playtime, median playtime, CCU etc. While SteamSpy offers valuable information for developers, researchers, and gamers, the API was not specifically designed to handle a high volume of requests at a rapid pace. The data is refreshed once a day, so there is also no reason to request the same information more than once every 24 hours. It accepts requests in a GET string and returns data in JSON arrays. All of the data is structured.

Data Collection Approach

Steam API is used to extract the entire list of available games on Steam, which is then used to scrape relevant data from other sources (SteamSpy, internal Steam API). Twitch does not allow for game title searches, so all live streams had to be extracted and later filtered for games of interest.

Steam and Steam API

Steam API has a limit of 1000 requests/hour, so proxies have been leveraged to get past rate limits.Using proxies, we've uploaded all Steam Curator reviews for target games as of February 2d, 2023. In addition to Steam API, this project involved web scraping using Scrapy to retrieve Steam reviews and Steam Curators data. The code parses the HTML responses received from the Steam website and extracts specific information about curators and their reviews in a structured manner.

Twitch API

Twitch API only allows retrieving streams that are live at the time of the query (no historical data is available). To get a good sample, a cron job has been leveraged to get all live streams every 20 min for 10 days (between January 31st and February 9th of 2023) It was inconvenient to save so many csv files so it was configured so that data gets written to the database directly. Once all Twitch streams data has been retrieved, additional requests were used to get data about broadcasters, their followers, users active in their chats etc.

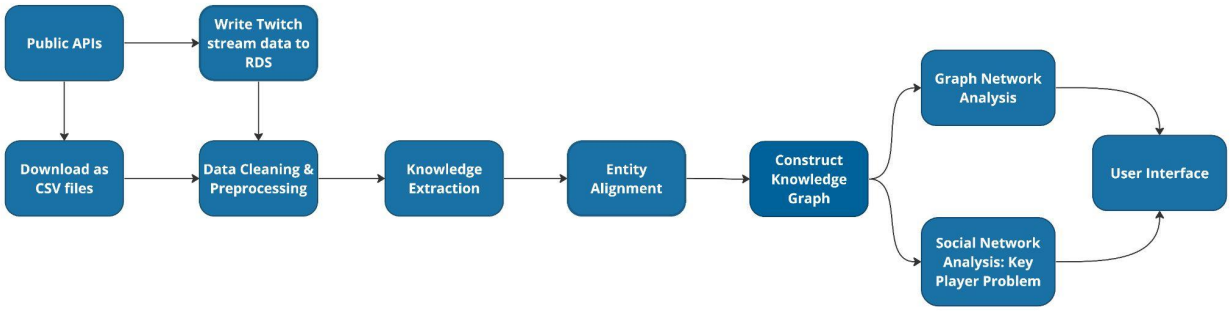
Data Sizes

Below is the summary of the data volume that has been extracted.

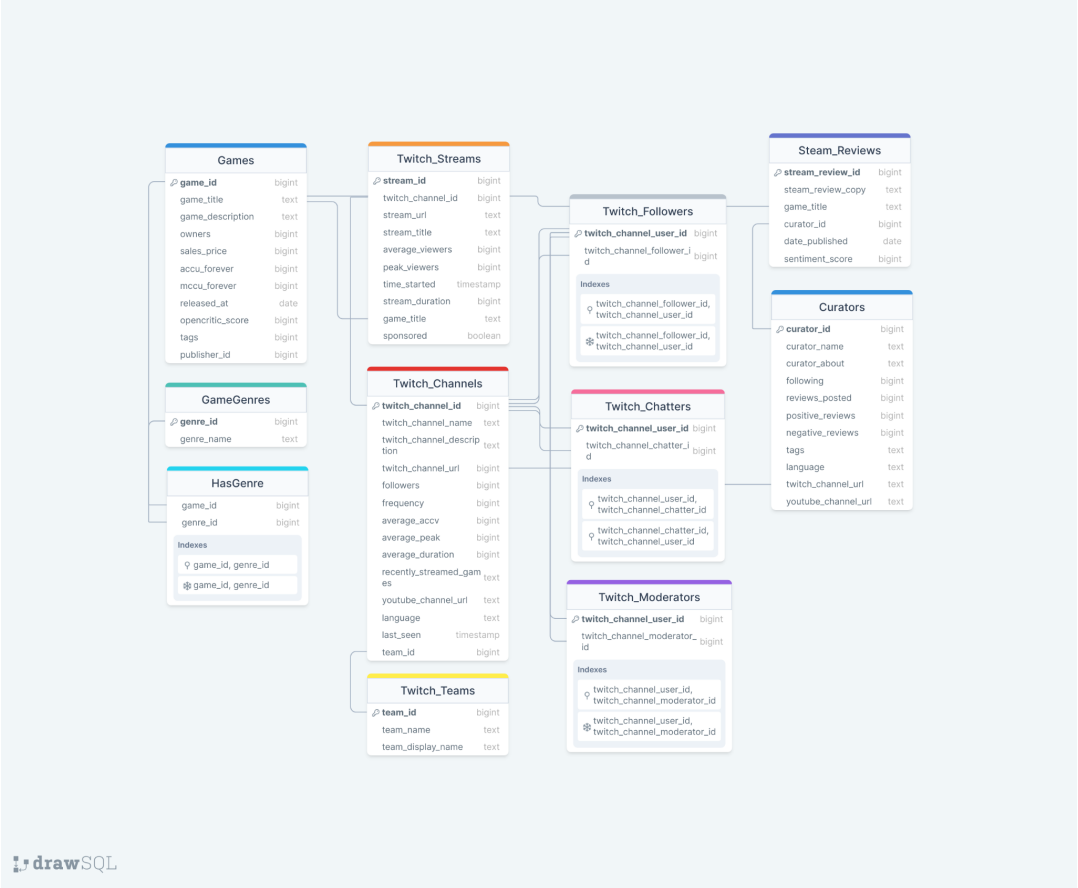
Source/Size	**Before Cleaning (raw)**	**After Cleaning**
Steam Games	154,750	72,234
Twitch Streams	4,974,151	163,780
Twitch Broadcasters	1,526,000	46,910
Twitch Users	6,228,377	2,609,307
Steam Reviews	942,826	757,424
Steam Curators	29,184	28,457

Data Pipelines

Below is the data pipeline of how the data was processed:



Although the majority of our processing was done using Jupyter notebooks hosted locally on personal computers, SQL queries have been also utilized directly in the database to perform some of the preprocessing steps. Once processed, the data was first stored in a relational Postgres Database on AWS RDS and accessed via PGAdmin4. Below is the database schema:



Then data was then processed and uploaded to Neo4j AuraDS. Neo4j AuraDS has been instrumental in the development of our MVP. Its completely managed, cloud-based infrastructure, made it easy to start using the Neo4j database in the cloud environment and allowed upscaling or downscaling the database resources based on application's needs. AuraDS also includes Graph Data Science Enterprise Edition for data science graph algorithms and Neo4j Bloom for visual data exploration.

Data Preparation

Quality Issues

Missing or incomplete data. Some streams lacked important information, such as game_id and game_title. A portion of Steam reviews didn't contain an actual text.

Different Languages. Since Twitch is a global platform, streams and chat messages can be in various languages.

Noise. One challenge is the presence of bot-generated content and streams where bots mimic real users. Additionally, the presence of typos, slang, non-english characters and emoticons in stream titles and Steam reviews makes the text less structured. Streams with a very low number of viewers may have limited significance for the research objective or advertisers. There is also a lot of non-gaming content on Twitch, which is not relevant for the purposes of this project.

Duplicate Data. The Twitch API provides access to a wealth of user data, including streamers, followers, chatters, and moderators. However, it was observed that there is a considerable overlap among these user categories, resulting in duplicate data.

Transformation

A set of steps was followed during the cleaning and preprocessing of the data. Firstly, attributes stored as JSON were unwrapped, and interesting attributes were flattened into columns or their own tables if they represented a multi-relationship. In the process of filtering streams, those with missing game_id/game_title were dropped. Streams with an average viewership of less than 20 were excluded as they were considered insignificant for advertisers. Non-English streams and non-gaming streams were also filtered out. For cleaning Steam reviews, those with missing copy and a length of less than 15 characters were removed. Reviews containing non-English characters were also filtered out. To ensure data integrity and avoid duplication biases, duplicate entries of Twitch users were identified and removed. Twitch bots were excluded from the analysis to eliminate their influence on the data. Text data, including Twitch stream titles and Steam reviews, underwent a cleaning process to remove noise such as typos, slang, and emoticons. Text normalization techniques were applied. To maintain consistency, the analysis primarily focused on English-language content so non-relevant languages were filtered out. In addition, entity matching was performed to link Twitch games with Steam games. Reviewers who were both Steam curators and Twitch streamers were also matched to enhance the dataset for analysis.

Features Selection

Two categories of feature sets were engineered: aggregate statistics of Twitch channels and text features of Twitch streams and Steam reviews. Channel performance was analyzed by examining the average reach and engagement within a recent time period as well as “top streamed games”. To mitigate the impact of outliers, the median was primarily utilized as the measure of central tendency. By analyzing the performance metrics of Twitch channel and understanding which games are most often broadcasted on the channel, advertisers can gain insights into the popularity of streamers within relevant gaming communities.

For the natural language processing tasks, the objective was to identify sponsored videos or streams. This was achieved by assigning the label 'Sponsored' to videos containing sponsorship-related keywords such as #ad or #sponsored. The open-source GPT language model was then employed to extract the names of sponsoring brands. Combining this knowledge with performance metrics, advertisers can pinpoint influencers who have already successfully engaged in sponsored content creation with their competitors.

In addition, sentiment analysis was conducted on review data from Steam Curators to assess the positive, negative, or neutral responses generated by a given game. By analyzing the sentiment of Steam curator reviews, advertisers can gain insights into the perception and opinion of different curators towards games similar to theirs.

Analysis Methods

a. Feature Generation

Identify Sponsors

In the context of sponsored streams, the identification of sponsored streams relies on whether the influencer receives payment for mentioning the sponsor brand. Streams containing strings like "Sponsored" or "#ad" are labeled as "Sponsored". The determination of the sponsors for these streams was also sought, aiming to gain insights into the current selection of influencers by brands for sponsorship and the evaluation of their outcomes. Although this information is not directly provided by the Twitch API, it can be inferred from the titles of the streams themselves.

To achieve this, three NLP models capable of performing named entity recognition (NER) tasks were utilized: the spaCy in-built pipeline NER, the flairNLP in-built pipeline NER, and the latest language models, such as GPT-3 and GPT-J. The models were assessed based on Precision and Recall scores above 0.8, considering a successful model. Notably, GPT-3 and GPT-J outperformed SpaCy and Flair significantly, leading to a focus on these two models for subsequent steps. The selected models for further analysis were text-davinci-003 and finetuned-gpt-neox-20b.

Steam Reviews Sentiment Analysis

Performing sentiment analysis on Steam reviews was important for building a knowledge graph around games and game reviewers. This information helps users understand the overall sentiment towards specific games and identify which games generate positive or negative responses. It also enables users to identify influential game reviewers related to their genre or similar games based on their sentiments.

The reviews were filtered to remove non-ASCII characters and then underwent text preprocessing steps such as lowercasing, removal of punctuation and stopwords, and stemming. Sentiment analysis was applied to the preprocessed reviews using the TextBlob library, which computes sentiment scores. These scores were transformed to a new range of 0 to 100. Polarity and subjectivity columns were added to the dataframe based on the sentiment scores.

b. Graph Analytics

There are 8 nodes and 11 relationship types. Overall, the graph consists of 3683643 nodes and 5274191 relationships with the following breakdown.

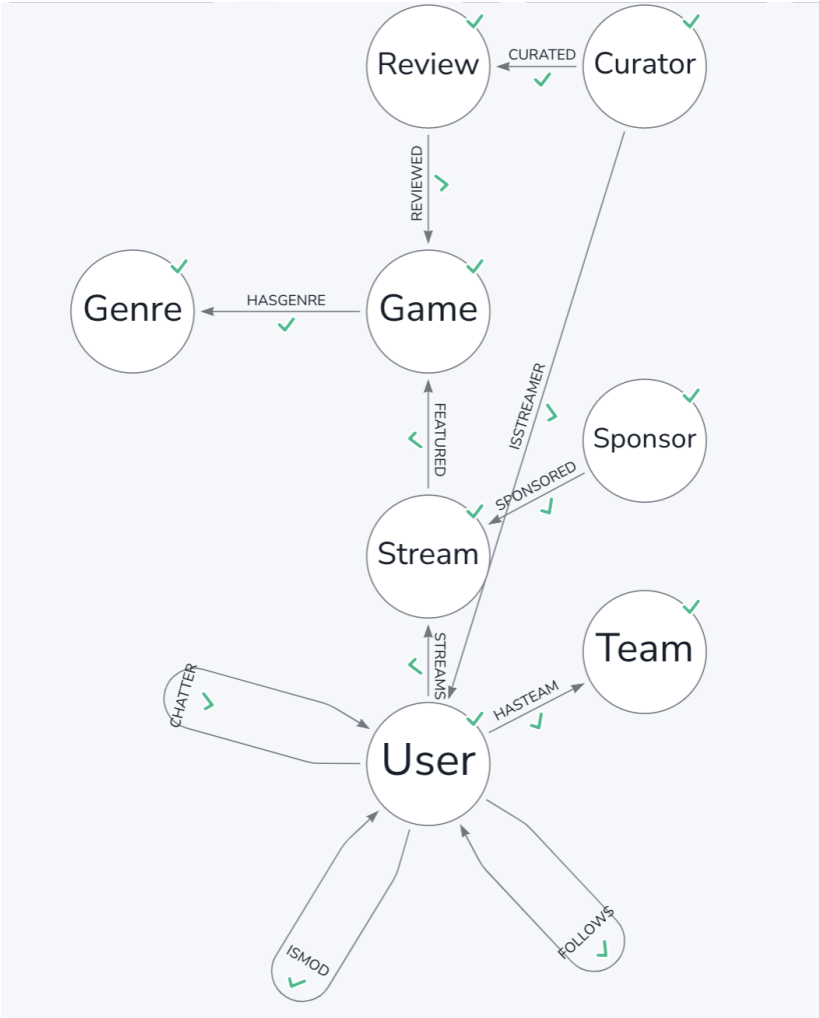
labels

```
{
  Game: 72233,
  User: 2656214,
  Streamer: 46910,
  Sponsor: 424,
  Curator: 29184,
  Stream: 163780,
  Team: 4348,
  Genre: 36,
  Review: 757424
}
```

The graph is composed of millions of Twitch users and their relationships between each other. Live streams of gameplay or activities are broadcasted by a small percent of Twitch users. Within the graph, users who broadcast live streams are labeled as Streamers. The teams they belong to (HASTEAM relationship) and the streams they broadcasted (STREAMS relationship) are also known. Additionally, the number of followers they had at the time of scraping (FOLLOWS

relationship), the users engaged in the streamer's chat (CHATTER relationship), and the role in which they participated are all known. It is possible to distinguish whether the user who chatted in the stream was a regular user (CHATTER relationship) or a moderator of the stream (MODERATOR relationship).

The graph also includes Steam curators, their reviews and relationship to Twitch users (ISSTREAMER relationship). Games belong to genres (HASGENRE relationship) and are featured in both Steam (FEATURED relationship) and Twitch Reviews (REVIEWED relationship). Sponsors are connected to streams (SPONSORED relationship).



To enhance the efficiency of the knowledge graph analysis, it was decided to opt to use AuraDS due to its graph data science capabilities and convenience. The compute cost of AuraDS can be dynamically scaled and managed, which was a key consideration in selecting AuraDS. The minimum package with 2 CPUs, 8GB of RAM and 16GB of storage is used for this project. To expedite the

process of uploading data, we utilized the latest low code solution, Data Importer. This tool allows users to import flat file data into Neo4j's graph database and model nodes and relationships visually mapping files to the model.

Once the knowledge graph was constructed, Cypher queries have been utilized to better understand the relationships in the graph. The queries were designed to analyze complex relationships between nodes and have provided deeper insights into the gaming ecosystem and its interdependent relationships. Examples:

1. Find games that are popular both on Steam and Twitch to identify twitch sponsorship opportunities for Steam game publishers.

To illustrate the way queries have been constructed, this query identifies games that have been reviewed on both Steam and Twitch, calculates the weighted average based on the ratio of Steam reviews to the total number of Steam reviews (weighted at 0.5) and the ratio of Twitch streams to the total number of Twitch streams (weighted at 0.5). The query helps determine the most popular games across both platforms, accounting for the different weights assigned to each platform

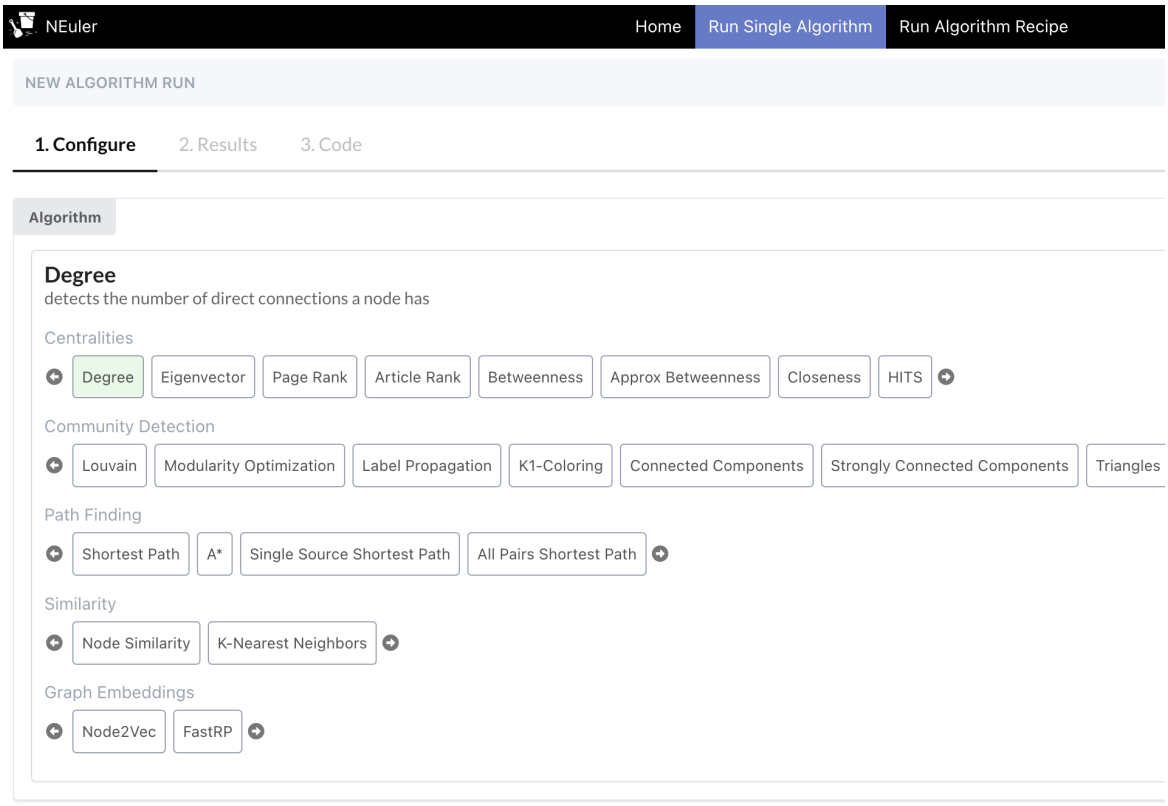
```
MATCH (game:Game)<-[:REVIEWED]-(steamReview:Review),
(game)<-[:FEATURED]-(twitchStream:Stream)
WITH game,
COUNT(DISTINCT steamReview) AS steamReviewCount,
COUNT(DISTINCT twitchStream) AS twitchStreamCount
WHERE steamReviewCount > 0 AND twitchStreamCount > 0

// Calculate total review counts for Steam and Twitch
WITH SUM(steamReviewCount) AS totalSteamReviewCount,
SUM(twitchStreamCount) AS totalTwitchStreamCount

// Join with the main query to calculate the weighted average
MATCH (game:Game)<-[:REVIEWED]-(steamReview:Review),
(game)<-[:FEATURED]-(twitchStream:Stream)
WITH game,
COUNT(DISTINCT steamReview) AS steamReviewCount,
COUNT(DISTINCT twitchStream) AS twitchStreamCount,
totalSteamReviewCount,
totalTwitchStreamCount

WHERE steamReviewCount > 0 AND twitchStreamCount > 0
RETURN game.game_title,steamReviewCount,twitchStreamCount,
(toFloat(steamReviewCount) / totalSteamReviewCount) * 0.5 + (toFloat(twitchStreamCount) /
totalTwitchStreamCount) * 0.5 AS weightedAverage
```

ORDER BY weightedAverage DESC

2. Compare streamers with highest number of chatter relationships with highest # of average viewers to understand relationship between engagement and popularity
3. Find Steam curators who are also active on Twitch and understand their preferences
4. Find the most popular game genres based on the number of Twitch streams, which could be useful for companies looking to sponsor Twitch streams, but don't have genre preference (non-gaming brands targeting gamers)
5. Find the top Twitch streamers who are also chatters based on their chat activity (they chat most or they chat in the popular streamers' chats) to identify potentially brand ambassadors
6. Find games that have been sponsored the most, along with their genres.
7. Identify similar Streamers based on the games they have featured on their streams. The Jaccard similarity coefficient was utilized. It provides a measure of how much overlap there is between the games featured by different streamers, indicating potential similarities in their content and audience preferences.

c. Social Network Analysis

For network analysis, Neo4j Graph Data Science (GDS) library was utilized. It comprises over 50 graph algorithms including community detection, centrality, and node embedding algorithms. By using Cypher projections, the stored graph in our database was projected to the in-memory graph format. To facilitate onboarding with GDS, The Graph Data Science Playground (NEuler) was leveraged. NEuler is a no-code UI that supports running each of the algorithms in the library, viewing the results, and providing the Cypher queries to reproduce the results.



To recap the structure of the user network, it should be noted that the data in the database was scraped between January 31st and February 9th of 2023, resulting in the presence of only those streamers who streamed over that period and users who chatted over that period. The network contains 3 types of relationships between users, namely follower, chatter, and moderator. Followers are users who show their interest in the content by following a streamer, while chatters are more actively engaged users who participate in conversations during the stream. As a result, this study focuses on analyzing the behavior of chatters. Streamers, who act as regular users, can follow other users, moderate other broadcasts, and engage in their chat, and are therefore represented as regular users with a secondary label Streamer, ensuring that no two nodes in the graph represent a single real-world entity.

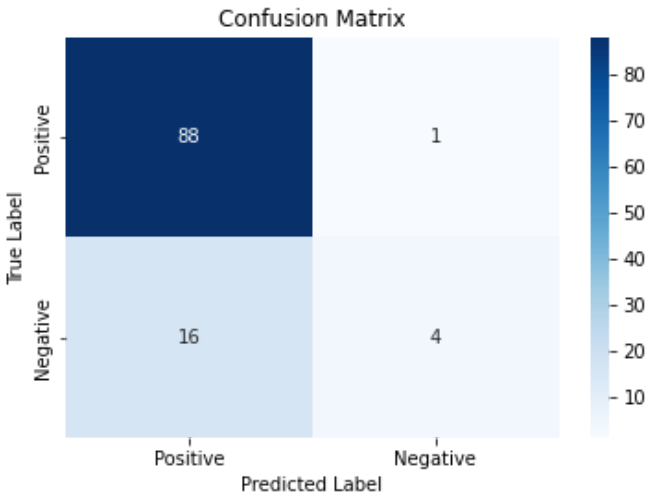
Centrality is a key property of complex networks that influences the behavior of dynamical processes and can bring important information about the organization of complex systems. There are many metrics to quantify the node centrality in networks or in the context of our research - influencing capability of a user in a social network. In this research we are focusing on In-Degree Centrality, Betweenness Centrality, Eigenvector Centrality, PageRank and Combined Centrality. Each centrality measure looks at the nodes influence from a different perspective and can be useful depending on analysis goals.

Findings & Reporting

a. Feature Generation

Identify Sponsors

In our training & validation set of 107 streams, we achieved precision of 0.87 and recall of 0.96 using GPT-J with the finetuned-gpt-neox-20b engine. For our test set of 109 streams, a precision of 0.82 and recall of 0.94 respectively, making our GPT model fairly accurate at deriving sponsors from the data.



	brand	precision	recall	f1-score	support
0	Xidax, Gamer Supps	1.000000	1.0	1.0	1
1	Xiae Edition	1.000000	1.0	1.0	1
2	Tacticus	1.000000	1.0	1.0	1
3	War Thunder	1.000000	1.0	1.0	2
4	MillionPugs	0.000000	0.0	0.0	0
5	Iron League	1.000000	1.0	1.0	1
6	The Great War Western Front	1.000000	1.0	1.0	2
7	Square Enix	1.000000	1.0	1.0	2
8	Sons Of Valhalla	1.000000	1.0	1.0	1
9	Rush.gg	1.000000	1.0	1.0	1
10	Rise of Kingdoms	1.000000	1.0	1.0	2
11	Raid Shadow Legends	1.000000	1.0	1.0	2
12	Poker	0.000000	0.0	0.0	1
13	Phyrexia	0.666667	1.0	0.8	2
14	Oodie, HelloFresh	1.000000	1.0	1.0	1
15	Oodie	1.000000	1.0	1.0	2
16	Novia	1.000000	1.0	1.0	1
17	NordVPN	1.000000	1.0	1.0	1
18	Moxfield	1.000000	1.0	1.0	1
19	Mech Arena	1.000000	1.0	1.0	10
20	Marvel, HelloFresh	1.000000	1.0	1.0	1
21	Marvel Snap	1.000000	1.0	1.0	3
22	Marvel	1.000000	1.0	1.0	4
23	Kingdoms Reborn	1.000000	1.0	1.0	1
24	Inkulinati	1.000000	1.0	1.0	1
25	Ikbibil	0.000000	0.0	0.0	0
26	HyperX	1.000000	1.0	1.0	2
27	HellpFresh	0.000000	0.0	0.0	0
28	HelloFresh, Skillshare	1.000000	1.0	1.0	1
29	HelloFresh	1.000000	1.0	1.0	27
30	Greenchef	1.000000	1.0	1.0	2
31	gPortal servers	1.000000	1.0	1.0	1
32	Gportal	0.000000	0.0	0.0	0
33	GGDrop	0.000000	0.0	0.0	0
34	Fortnite	1.000000	1.0	1.0	1
35	Factor	1.000000	1.0	1.0	4
36	Corsair, Raid Shadow Legends	1.000000	1.0	1.0	1
37	Blacksail	1.000000	1.0	1.0	1
38	Babbel	1.000000	1.0	1.0	4

Based on 2167 Sponsored Streams:

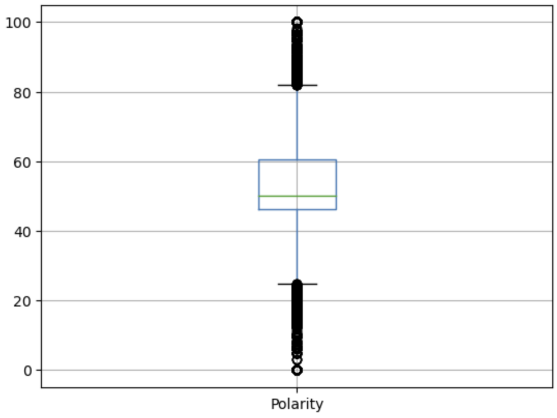
- Non-Gaming Sponsors, **specifically prep meal delivery** are dominating the space. Top non-gaming sponsors based on the # of sponsored streams: **HelloFresh, Factor, Green Chef (all prep meal delivery brands)**
- Sponsors who gained most average views (in total): **HelloFresh, Gillette, Nitro**

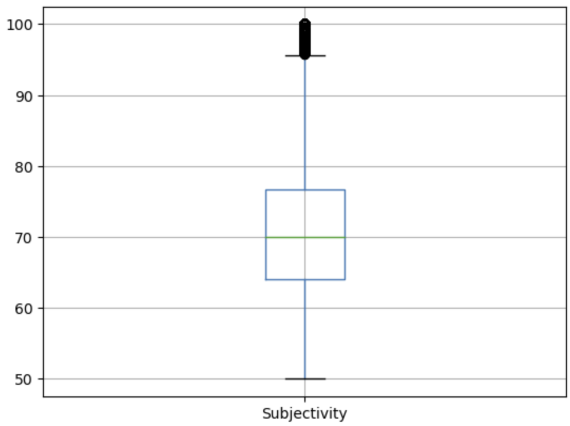
- When it comes to gaming brands, mobile game publishers are most active. Top gaming sponsors based on the # of sponsored streams: **Mech Arena, Fortnite, Rise of Kingdom (all mobile games)**
- Top 3 games popular among top gaming sponsors based on total # of average viewers: **SMITE, GTA5, Fortnite**
- Top 3 games popular among non-gaming sponsors based on total # of average viewers: **Dead by Daylight, Hearthstone, Teamfight Tactics**

This is a very useful result, as it is indicative that we have at our disposal a modeling system to rapidly derive sponsors in semi-supervised streaming data.

Steam Reviews Sentiment Analysis

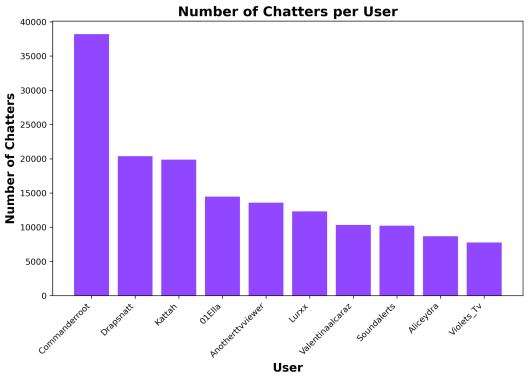
Games got an average Polarity Score of 52.77 and an average Subjectivity Score of 69.54.

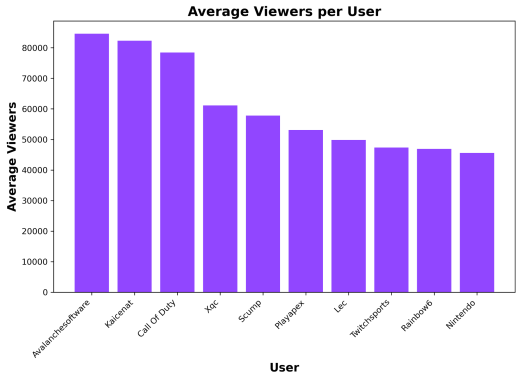




b. Graph Analytics - Highlights

1. Interestingly, there is no overlap in top 10 Streamers with most average viewers and most chatters, which means streamers that have high viewership do not necessarily have high engagement.





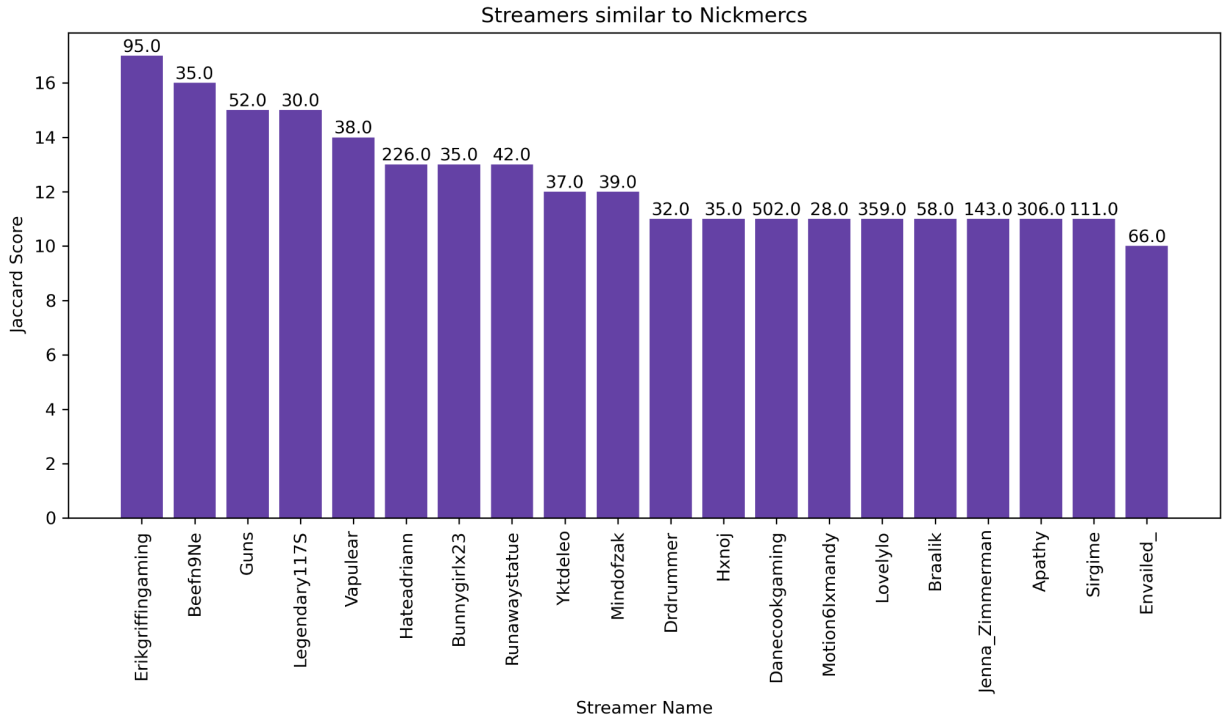
2. Here are the top 10 games popular on both platforms:

	game.game_title	steamReviewCount	twitchStreamCount	weightedAverage
0	Quell Zen	4	464	0.186963
1	Call of Duty: World at War	551	6	0.149181
2	Call of Duty: Modern Warfare 2 (2009)	371	4	0.100430
3	The Ship: Murder Party	269	14	0.077266
4	D: The Game	10	134	0.056350
5	Astral Heroes	2	113	0.045805
6	Men of War	37	87	0.044712
7	HOMEBOUND	3	64	0.026440
8	Spikit	1	59	0.023904
9	Intruder	78	2	0.021579

3. Steam curators who are also active on Twitch prefer the following genres: Action, Massively Multiplayer, Racing, Indie Strategy. Warhammer is the most frequently mentioned game by reviewers with strong followings on both Steam and Twitch.

	curatorName	steamFollowers	averageTwitchViews	totalfollowing	topGames	genres
0	Warhammer	253488	356.0	253844.0	[Warhammer: Vermintide 2, Total War: WARHAMMER...	[Action, Massively Multiplayer, Racing, Indie,...
1	IGN	180928	303.0	181231.0	[Warhammer 40,000: Gladius - Relics of War, Pl...	[Action, RPG, Sports, Indie, Adventure, Design...
2	Yogscast Games	141743	476.0	142219.0	[The Sinking City, Zombotron, Pillars of Etern...	[Indie, Adventure, Casual, Action, Racing, Uti...
3	Builders, managers & commanders	120378	30.0	120408.0	[Mini Metro, The Innsmouth Case, Unholy Heights]	[Adventure, Strategy, Racing, Indie, Action, S...
4	Extra Credits	103680	78.0	103758.0	[Brothers - A Tale of Two Sons, Sonic Mania, O...	[Adventure, Free to Play, Indie, Game Developm...
5	Vinesauce Vidya	94334	6665.0	100999.0	[Killing Floor 2, Planet Coaster, I Am Bread]	[Indie, RPG, Action, Early Access, Free to Pla...
6	WGN Chat	42223	39.0	42262.0	[The Disney Afternoon Collection, Stick Fight:...	[Indie, Action, Simulation, Adventure, Casual,...
7	The AngryJoeShow Army =AJSA=	38200	821.0	39021.0	[OnlyCans: Thirst Date, Far Cry 4, Divinity: D...	[Simulation, Education, Adventure, Action, Ind...
8	Giant Bomb Staff	30056	513.0	30569.0	[Marvel Heroes Omega, Divinity: Original Sin 2...	[Adventure, Indie, Strategy, Casual, RPG, Simu...
9	PsiSyndicate	28682	63.0	28745.0	[Project Zomboid, The Mortuary Assistant, Last...	[Sports, RPG, Simulation, Indie, Action, Web P...

4. Nickmercs is one of the most popular streamers on Twitch with 6.6M followers and 20K average viewers. Here is the top 20 streamers most similar to him based on the games they play (using Jaccard similarity)



c. Social Network Analysis

In-Degree Centrality

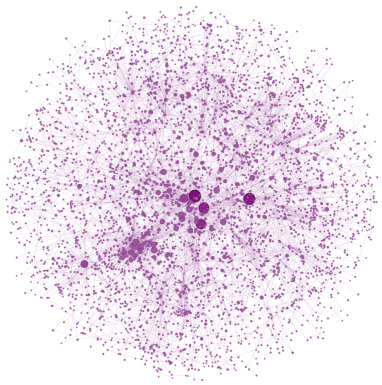
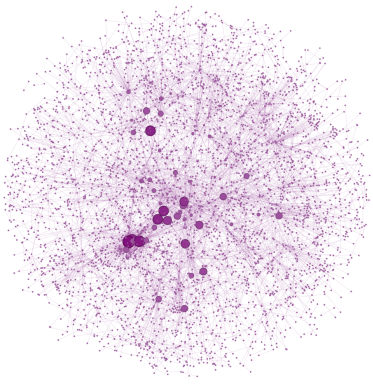
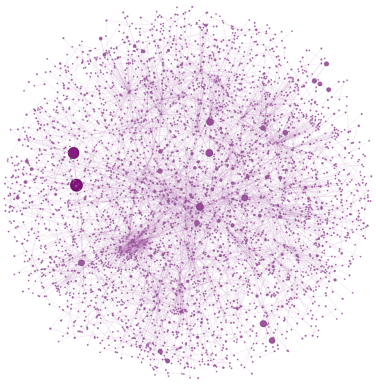
In-degree centrality measures the number of incoming relationships (chatter messages) directed towards a Twitch user. It indicates how many other users are engaging with and chatting during the streams of the user. In-degree centrality is a measure of popularity and, more importantly, engagement. Twitch users with high in-degree centrality have a large number of other users actively engaging and chatting during their streams. This indicates not only the level of attention they receive from the Twitch community, but also the level of interaction, discussion, and engagement happening around the user's content.

Betweenness Centrality

Betweenness Centrality calculates the extent to which a node (in our case, a Twitch Streamer) lies on the shortest paths between other nodes in the network. In the context of Twitch chat engagement, a Streamer with high betweenness centrality would act as a bridge or intermediary, connecting different segments of the network. This indicates that the user's chat messages are reaching a diverse set of streamers and viewers, potentially exerting significant influence. By participating in chat conversations with multiple users, they create a bridge that enables communication and interaction between otherwise disconnected users.

PageRank

PageRank is also a useful centrality measure for evaluating the influence of Twitch users in the network based on chat engagement. In the context of Twitch chat engagement, PageRank can identify Twitch users who receive chat messages from other influential users. This suggests that the user has a significant presence and influence in the Twitch community.

In-Degree Centrality	Betweenness Centrality	PageRank
		

Combined Centrality

Different centrality measures capture different aspects of node importance. Combining two or more centralities can help gain a more comprehensive understanding of influence dynamics and identify top influencers more accurately. Based on the research in the field, the combination of a local centrality measure with a global centrality measure proves to be particularly effective in identifying the most influential nodes. The addition of a global centrality indicator helps guide the prediction towards more central regions in the network, complementing the rankings provided by local centrality measures.

For this project, in-degree centrality has been combined with eigenvector centrality. The combined score is calculated as a weighted average of the degree centrality score and the eigenvector centrality score. The formula for the combined score is:

CombinedScore = (0.5 * degreescore) + (0.5 * eigenvectorscore)

The reason for calculating the combined score is to consider multiple aspects of a streamer's influence within the Twitch network. Degree centrality measures the number of connections a streamer has, representing their popularity and reach. On the other hand, eigenvector centrality takes into account the influence of the streamer's connections, considering both the number and quality of connections they have.

By taking a weighted average of both scores, with equal weight given to each, the combined score aims to provide a balanced representation of a streamer's overall influence. This approach acknowledges that a streamer may have a high degree centrality, but low influence from their connections, or vice versa. The choice of equal weighting is subjective and can be adjusted based on the specific marketing campaign requirements or the context of the analysis.

Evaluation

Gaining accurate ground truth for the influence of nodes in the Twitch network is a crucial but challenging task. Several approaches can be utilized to achieve this goal effectively.

- Firstly, analyzing historical data can assist in understanding the past performance and impact of streamers. By examining viewership numbers, engagement metrics, and audience demographics, insights can be gained into the influence of streamers in terms of reach and engagement.

- Another approach is conducting surveys and interviews with Twitch users to gather their perceptions and opinions regarding influential streamers. This qualitative data can provide valuable insights into the reputation, credibility, and impact of streamers within the Twitch community.

- In addition to historical data analysis and user opinions, expert evaluation and industry knowledge can significantly contribute to determining influential streamers. Experts in the gaming and Twitch industry possess valuable insights into trends, emerging talent, and the dynamics of influencer marketing. Their subjective judgments, combined with data-driven analysis, can provide a more comprehensive understanding of the most influential nodes in the Twitch network.

Reporting

The end product of the project is **GamerGraph** - a solution that demystifies dynamics of gaming communities and provides video game publishers with accessible, data-driven tools to identify key players within these communities.

The goal of the reporting dashboard is to provide relevant and meaningful information that effectively communicates the findings to the intended audience. The dashboard aims to assist users in identifying key players and influencers in the gaming industry based on various criteria such as game title, genre, key competitors, and target influencers.

In deciding what to present in the reporting dashboard, several factors were taken into consideration based on the challenges our target audience faces.

- Understand the audience. The audience, including the Marketing & PR team or other decision-makers involved in the promotional strategy, was taken into consideration. To begin the process, a user persona was constructed based on the characteristics and responsibilities of the target audience. In this case, the user persona of Ann, a Marketing Analyst at TinyBuild Games, was created.
- Define goals. The persona encapsulates audience' goals, challenges, and objectives in identifying Twitch Streamers. Additionally, a case study was developed, focusing on the specific scenarios faced by Ann while promoting a new game during different stages of its release.
- Prioritize relevant metrics. Developing persona and case study provided insights into the metrics Ann considered, such as viewership, relevance, similarity and influence within the Twitch community.

User Interaction (Flow)

Based on the performed analysis, the following user flow has been developed.

Step 1: Gathering User Input The user begins by answering a series of questions about their game, including its title, genre, and key competitors. Additionally, the user provides information about their target influencers, specifying the desired reach and providing examples of "ideal" influencers

Step 2. Twitch Streamer and Steam Curators Stats. Upon receiving the user's input, the system retrieves relevant data from Twitch Streamers and Steam Curators. The returned data is presented in the form of charts and tables, which can be filtered based on various criteria such as genre, competitors, brands' previous mentions, and similar influencers. Users can sort columns based on different values and export the data to a CSV file.

Step 3. Key Players in the Network. The system generates a subgraph of Twitch Streamers, highlighting the key players based on centrality measures. Users have the flexibility to filter the subgraph based on genre, competitors, brands' previous mentions. The analysis can be performed using different centrality measures, such as PageRank, Degree, Eigenvector, or a combined centrality measure. Additionally, users can explore the relationships between influencers by zooming in and out of the graph, gaining further insights into their network connections.

Techniques and Tools for Communicating Results

User Interface

Figma: the UI design process started with creating a user experience (UX) design in Figma. It allowed for creating wireframes and visualizing the app's layout, interactions, and overall flow. The wireframes in Figma showcased the structure of the page, including the header with tabs and filters, as well as the two distinct panels for the chart area and table area. These wireframes served as a visual representation of the intended UI design.

CSS: CSS (Cascading Style Sheets) was used to apply styles and layout to the UI components. CSS ensured consistent and visually appealing presentation of the elements, including font styles, colors, spacing, and positioning. It helped in achieving a polished and professional look for the user interface.

UIKit-to-SCSS Figma Plugin: The uikit-to-scss Figma Plugin was employed to export SCSS (Sass) files containing the styles of UI elements from Figma. This plugin streamlined the process of transferring the design styles into code. By exporting the SCSS files, the predefined styles have been incorporated into the React components, ensuring visual consistency throughout the application.

React with TypeScript: The layout design was implemented using React, a JavaScript library for building user interfaces, in combination with TypeScript. React provided a component-based architecture that allowed for the creation of modular and reusable UI components. TypeScript, being a typed superset of JavaScript, added static typing, improving code quality and maintainability. React with TypeScript enabled efficient development and ensured robust UI components for the building layout.

Recharts: To create visually compelling charts for displaying data related to Twitch Streamers and Steam Curators, the Recharts library was employed. Recharts is a composable charting library built

on React components. It offers a variety of customizable chart types, such as line charts, bar charts, and pie charts. By utilizing Recharts, the project could present data in an engaging and informative manner, enhancing data visualization capabilities.

Integration with Neo4j

Use-neo4j Hooks: The app leveraged the use-neo4j hooks, which provided a convenient interface for sending parameterized Cypher queries to the Neo4j database. Specifically, the ConnectDatabase and ExecuteQuery components were utilized to establish the connection with the Neo4j database and execute queries respectively. These hooks abstracted the complexity of interacting with the database and simplified the process of retrieving data.

Redux: Redux was employed to manage shared state within the application, particularly for handling form data that needed to be accessed across multiple pages. The Redux store was utilized to hold the application's state and define a slice dedicated to managing the form data. Actions were created to update the state, ensuring that the form data remained consistent and accessible throughout the application.

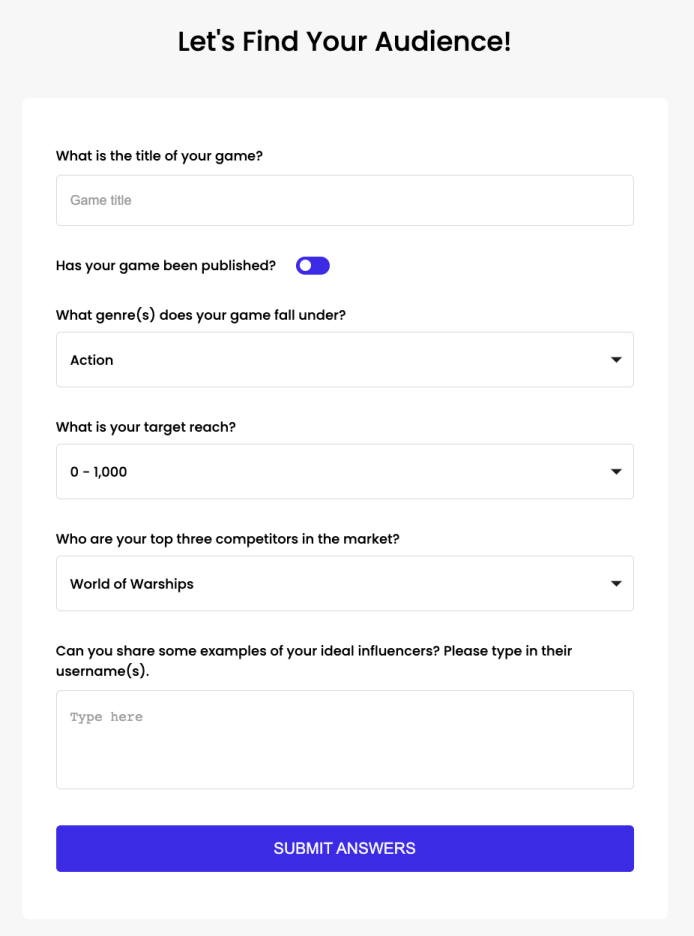
Cypher Queries with Parameters: these queries allowed for dynamic and flexible data retrieval from the Neo4j database based on user inputs in the form. Parameters act as placeholders that are populated with actual values provided by the user at runtime

Vercel: In addition to the above tools, the application also utilized Vercel for previewing the application. By integrating Vercel into the development workflow, the application could be tested and reviewed in a controlled environment before being deployed to production.

Reporting dashboard

User Interaction (Flow)

Step 1: Gathering User Input



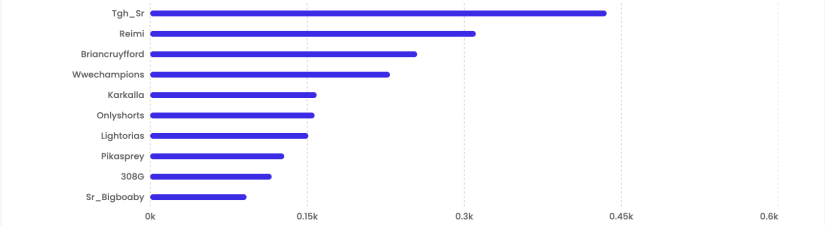
Step 2. Twitch Streamer and Steam Curators Stats

Results

Twitch streamers | Steam curators | Network

Top Twitch Streamers by Viewers (64)

Filter by: Genre



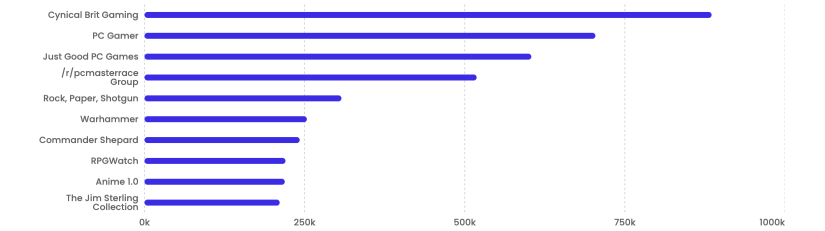
Streamer name	URL	Following	Avg. views	Peak views
Tgh_Sr	https://www.twitch.tv/tgh_sr	58,642	436	579
Reimi	https://www.twitch.tv/reimi	3,602	311	562
Briancruyfford	https://www.twitch.tv/briancruyfford	14,715	255	374
Wwechampions	https://www.twitch.tv/wwechampions	9,061	229	298
Karkalla	https://www.twitch.tv/karkalla	5,267	159	186

Results

Twitch streamers | Steam curators | Network

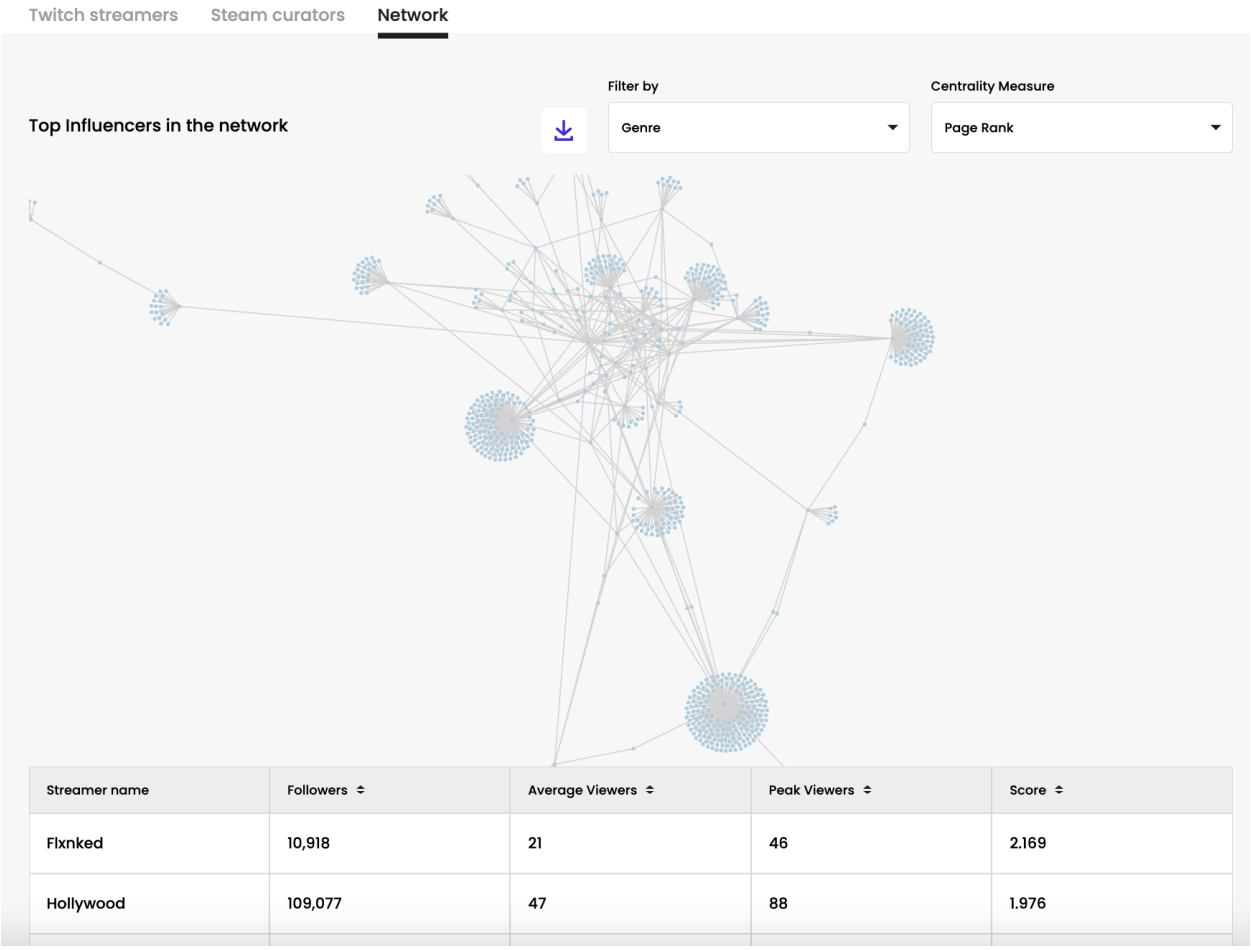
Top Steam Curators by Followers (24573)

Filter by: Genre



Curator name	URL	Following	Reviews Posted	Recommended	Not Recommended
Cynical Brit Gaming	https://store.steampowered.com/curator/1370293-Cynical-Brit-Gaming/	885,366	206	204	2
PC Gamer	https://store.steampowered.com/curator/1850-PC-Gamer/	703,968	408	408	0
Just Good PC Games	https://store.steampowered.com/curator/6856269-Just-Good-PC-Games/	603,914	390	390	0

Step 3. Key Players in the Network



Solution Architecture

Requirements & Metrics

1. **Up-to-date data.** The system should provide access to data that is up to date, providing the most accurate and current information possible. Access to the up- to-date data is crucial because it allows video game publishers to make informed decisions based on the most current and accurate information available. How to measure:

- **Data freshness** measures how recently the data was updated in the system. It can be measured by comparing the timestamp of the latest data update with the current time.
- **Latency** measures the time it takes for the data to be updated in the system and made available to users. Low latency is critical for ensuring that users have access to the most up-to-date information as quickly as possible.

2. **Speed.** The system should provide fast response times for queries and data updates to ensure a good user experience. Fast response times are important to ensure a good user experience. Slow response times can lead to frustration and can deter users from using the system. How to measure:

- **Query response time** measures time it takes for the system to respond to user queries
- **Throughput** measures the number of requests that can be processed by the system in a given time period.

3. **Scalability.** The system should be able to handle a growing volume of data from new sources, such as YouTube, Reddit, or Discord etc. Scalability is important to ensure that the system can handle growing volumes of data as new sources are added, and to prevent performance issues as the system grows. How to measure:

- **Scalability testing** which involves testing the system under different conditions to simulate a growing volume of data and requests. This can help identify any bottlenecks or performance issues that need to be addressed to ensure scalability.

4. **Cost.** The system should be optimized for cost to ensure that it is affordable to maintain and scale. High costs can make the system unaffordable to maintain and scale, which can limit its usefulness and impact. How to measure:

- **Cost per transaction** measures the cost of processing each individual transaction or query in the system.
- **Infrastructure and operational costs** measure the cost of the infrastructure required to support the system as well as the ongoing costs associated with operating the system.

5. **Availability.** The system should be highly available and should be able to handle high traffic loads without any downtime. High availability is essential to ensure that the system is accessible to users at all times. Downtime can lead to lost opportunities and decreased user confidence. How to measure:

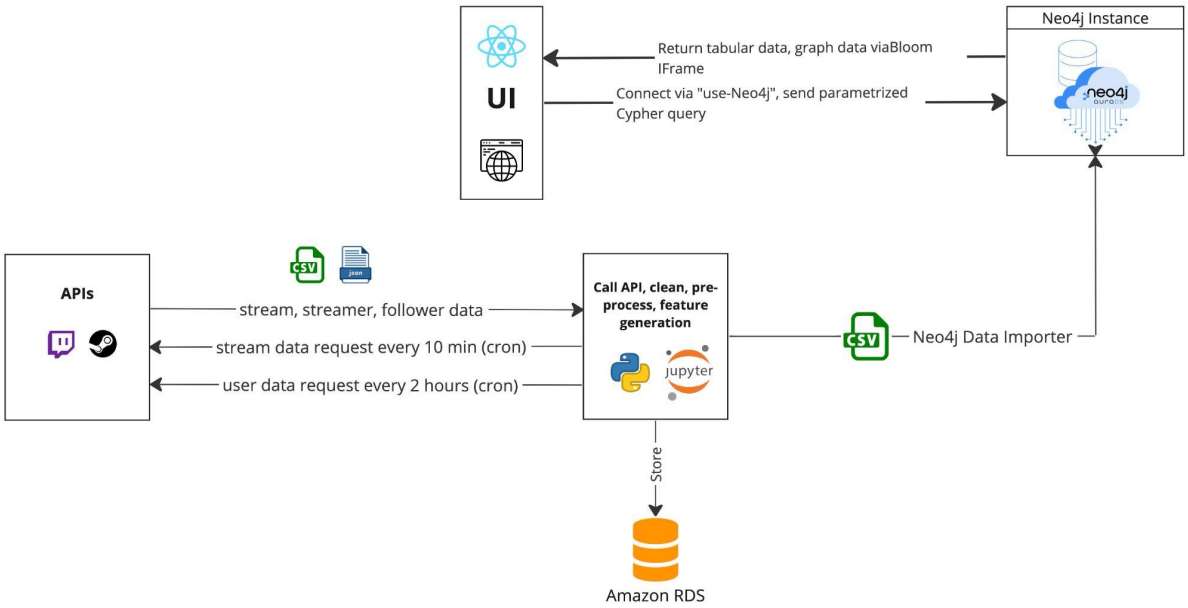
- **Uptime** measures the percentage of time the system is available and accessible to users. A high uptime percentage indicates high availability and reliability.

Case Study

Goal: to scale an existing web app that collects data from Steam and Twitch APIs, preprocesses it, and stores it in a Neo4j graph database. The app will be scaled to add support for new data sources, such as YouTube and Twitter, and to increase the amount of data processed from existing data sources. The system should be able to ingest and process new data without compromising performance and should be optimized for cost.

Adding new data sources and processing more data from existing sources can lead to an increased data volume that may result in performance degradation. Ingesting data from multiple sources requires careful management of API rate limits, data formats, and data quality issues. Preprocessing data from multiple sources and integrating it into the existing system can be complex and time-consuming, and it is critical to clean, process and integrate the data effectively for maintaining data quality and system performance. Additionally, as the system scales, costs for storage and processing can increase significantly, so it is also crucial to optimize the system for cost to maintain profitability and scalability.

Current Data Pipeline



Scalability Approach

1. Data extraction and pre-processing

Problem: Relying on a local machine for preprocessing imposes limitations on processing power, memory, and storage capacity. Manually feeding CSV files from one task/process to the next is time-consuming, prone to errors, and challenging to manage. Using cron jobs to send regular API requests can create redundant or irrelevant data.

Solution: To address these issues, the system needs to be optimized by moving data processing to cloud-based services and using event-driven architecture instead of cron jobs to retrieve data from API providers. Also, Twitch API responses can be cached to reduce the number of requests sent to the server.

2. Data storage

Problem: The current approach of using AWS RDS for intermediary data storage makes the system relatively expensive to operate and difficult to scale as the system and the volume of data grows.

Solution: uploading pre-processed data directly to Neo4j and using AWS S3 or a data lake for raw data storage can eliminate these inefficiencies. This approach can provide a more scalable and cost-effective solution for handling large volumes of data.

3. Neo4j AuraDS

Problem: Neo4j AuraDS has been instrumental in the development of our MVP. Its completely managed, cloud-based infrastructure, made it easy to start using the Neo4j database in the cloud environment and allowed upscaling or downscaling the database resources based on application's needs. AuraDS also includes Graph Data Science Enterprise Edition for data science graph algorithms and Neo4j Bloom for visual data exploration. However, Neo4j AuraDS is available only on Google Cloud (Early Access Program available for AuraDS on AWS as of March 2023) and is costly ($0.125 USD per GB/hour For running instances and $0.025 USD per GB/hour for paused instances with a minimum instance of 8GB)

Solution: Depending on the project requirements and funding, it might be worthwhile to switch either back to Neo4j Community or Neo4j Enterprise.

4. Neo4j Data Upload

Problem: Manual process of uploading data to Neo4j is time-consuming and not suitable for large amounts of data. As the amount of data increases, this process can become unmanageable and result in delays.

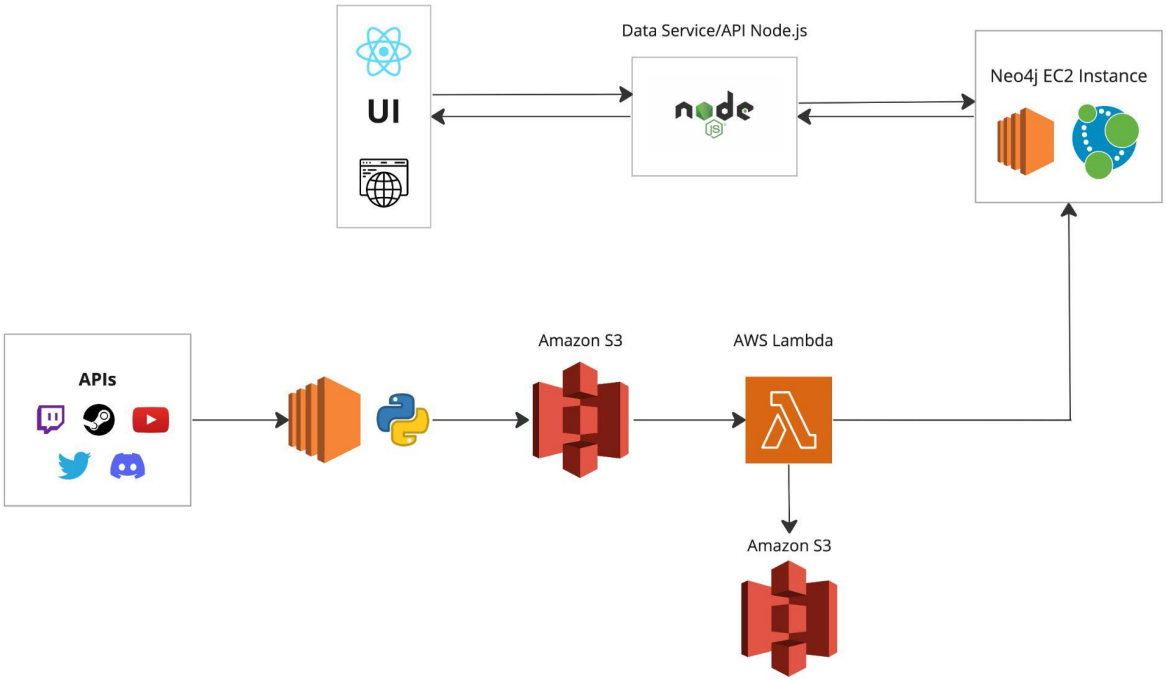
Solution: automate the process by using a data pipeline tool that can automatically load the data into Neo4j. This tool can monitor the API for new data and trigger the pipeline to upload the data to Neo4j automatically.

5. User Interface - database connection

Problem: Sending queries directly from the user interface to the Neo4j database can introduce security risks, increase the load on the database and cause performance issues, especially if many users are submitting queries simultaneously. Also, it can make it difficult to manage and monitor database access as it would require managing user permissions and monitoring query usage.

Solution: implement an API layer between the user interface and Neo4j database. This API layer can serve as a gatekeeper and perform validation on incoming queries, handle user authentication and authorization, as well as provide a centralized point for monitoring and managing query usage, allowing for easier tracking and troubleshooting of issues.

Proposed Architecture



Conclusions

This data science project aimed to achieve two main goals: organizing data into a knowledge graph and identifying key players in the network of gaming reviewers. Throughout the project, several important discoveries and challenges were encountered.

When building the knowledge graph, matching entities and determining which data sources to include in the knowledge graph are time-consuming tasks that require careful consideration. Additionally, it was found that having a scalable and automated backend system is crucial. It needs to be noted that the main focus was on the data extraction and analysis, and the scalability issues were not addressed. Consequently, the product is not yet ready for deployment, and a scalability plan needs to be implemented to handle the high volume of data. For example, accessing the necessary statistics from the Twitch API posed a challenge, as historical data is not available, and frequent updates are required.

In terms of identifying key players, a major challenge was encountered in validating the results. Further analysis is needed to understand the past performance and impact of streamers by analyzing historical data. This additional analysis will help refine the identification process and provide a better understanding of the network dynamics.

In conclusion, this data science project provided valuable insights into the complexities of organizing data into a knowledge graph and identifying key players in a gaming reviewer network. Ongoing analysis and validation will be essential to improve the accuracy and depth of the project's results.

Appendices

DSE MAS Knowledge Applied to the Project

The application of DSE MAS knowledge to this project was crucial. The primary programming language used for the project was Python, and the course "Python for Data Analysis" played a significant role in enabling me to effectively develop and iterate on the source code. The knowledge acquired from the Data Integration and ETL course proved invaluable. This course equipped me with the skills to efficiently extract and transform data and introduced me to the power of knowledge graphs. Moreover, the course on Beyond Relational Data Models expanded my understanding of Neo4j and familiarized me with working with Cypher queries. This knowledge deepened my ability to navigate and manipulate the data stored in Neo4j, enhancing the project's overall data management and analysis capabilities.

Link to the Library Archive for Reproducibility

TBP