ISSN 2723-6471

Efficient Web Mining on MyAnimeList: A Concurrency-Driven Approach Using the Go Programming Language

Muhammad Daffa Arviano Putra[1], Deshinta Arrova Dewi[2,*], Wahyuningdiah Trisari Harsanti Putri[3], Harry Tursulistyono Yani Achsan[4]

[1,3,4]*Department of Informatics-Faculty of Engineering Science, Paramadina University, Jakarta, Indonesia*

[2]*Faculty of Data Science and Information Technology, INTI International University, Nilai, Malaysia*

(Received: July 14, 2024; Revised: August 24, 2024; Accepted: September 12, 2024; Available online: September 23, 2024)

Abstract

Anime is a globally popular form of entertainment, with the industry experiencing rapid growth in recent years. Despite the wealth of anime data available on MyAnimeList, the largest community-driven platform for anime enthusiasts, existing publicly available datasets are often outdated and incomplete. This presents a challenge for data science research, as the increasing volume of anime information requires more efficient data extraction methods. This research aims to address this challenge by developing a concurrent web mining program using the Go programming language. Leveraging Go's concurrency capabilities, our program efficiently extracted anime data from MyAnimeList, iterating through anime pages from ID 1 to 52,991. To overcome potential issues like rate limits and server timeouts, we implemented a two-phase execution strategy. As a result, the program successfully gathered 23,105 anime records within 8.5 hours. The extracted data has been transformed into a comprehensive dataset and made publicly available in CSV format. This research demonstrates the effectiveness of concurrent web mining for large-scale data extraction and offers a valuable resource for future data-driven research in the anime industry.

Keywords: Concurrent Web Mining, Anime Data Extraction, Go Programming Language, Myanimelist Dataset, Data Science in Anime Industry, Process Innovation

1. Introduction

Anime, a distinctive form of animated entertainment originating in Japan, has garnered a massive global audience due to its diverse storytelling, unique visual style, and wide-ranging genres [1]. Over the years, anime has transitioned from a niche interest to a mainstream cultural phenomenon, with each season—winter, spring, summer, and fall—bringing forth a wealth of new titles. The anime industry continues to grow rapidly, both in Japan and internationally, with its market value reaching 20 billion USD in 2021 [2], [3]. As part of the entertainment landscape, anime holds a particularly strong appeal among younger generations, and nearly 300 new anime titles are released annually to meet the demand of eager fans [4], [5]. Beyond entertainment, enthusiasts actively engage with anime not just for enjoyment but to track the ongoing trends and developments within the industry itself.

However, the rapid growth of the anime industry has led to challenges in navigating and analyzing the vast amount of data associated with it. MyAnimeList, the largest community-driven platform for anime enthusiasts, has emerged as a critical resource [6]. It serves as a comprehensive hub for anime information, offering users a space to share opinions, reviews, and recommendations [7]. With over 50,000 unique anime titles listed, MyAnimeList represents an invaluable data repository, reflecting the immense enthusiasm and activity of the global anime community. This platform, therefore, holds substantial potential for data science research aimed at exploring trends, popularity, and other critical aspects of the anime industry.

Despite the vast potential of MyAnimeList for data-driven research, extracting and utilizing this wealth of information remains a complex challenge. Manual data extraction is inefficient and time-consuming, given the sheer volume of content available. To harness the full potential of this data, it is necessary to implement a more efficient technique to

***Corresponding author: Deshinta Arrova Dewi (deshinta.ad@newinti.edu.my)**
DOI: https://doi.org/10.47738/jads.v5i3.352

retrieve large amounts of data automatically. Web mining offers a solution by enabling the systematic collection of data from websites [8]. However, due to the scale of MyAnimeList's content, traditional web mining techniques—if performed sequentially—would require significant time and computational resources. Thus, a more efficient approach is needed. This research aims to introduce a faster, more efficient approach to web mining, specifically designed to handle large datasets like those on MyAnimeList. By leveraging the concept of concurrency, this study proposes a solution that allows data to be retrieved from multiple web pages simultaneously. The primary focus is to implement this concurrent data retrieval using the Go programming language, which is well-suited for concurrent tasks due to its lightweight goroutines. This research will demonstrate how the application of concurrency can significantly accelerate the process of web mining and data extraction from MyAnimeList.

The proposed solution is not only relevant to the growing anime industry but also contributes to advancements in web mining methodologies. By improving the efficiency of data retrieval, this research provides a foundation for further studies that rely on large-scale datasets. Moreover, the structured dataset generated from the extracted anime data will be a valuable resource for researchers and developers in the data science field. In essence, this work addresses a critical challenge in web mining and offers a scalable solution that can benefit future studies of the anime industry and other large-scale online platforms.

To achieve the objectives of this research, we will implement a web mining program that retrieves anime data concurrently using Go's concurrency features. This approach enables the program to send requests to multiple pages on MyAnimeList at the same time, significantly reducing the time required for data extraction. The output of this process will be a structured dataset containing thousands of anime titles, which can be used for various data science applications.

2. Related Works

Previous research on efficient web mining methods has contributed significantly to the advancement of large-scale data extraction techniques. For instance, [9] introduced an innovative strategy that utilized a hundred threads within a single web crawler, resulting in a substantial increase in web mining speed. This approach has been foundational, inspiring further research that explores the potential of multi-threading in web mining, especially for large-scale data retrieval.

In addition to this, [10] demonstrated the efficacy of multi-threading techniques specifically in mining real-time sports news data. The study showed a considerable improvement in mining speed and overall efficiency, further validating the advantages of multi-threaded approaches in handling time-sensitive data extraction tasks.

The limitations of single-threaded web crawlers have also been well-documented. Research in [11] observed that single-threaded crawlers significantly prolong the data extraction process due to their inability to achieve true concurrency. The sequential nature of single-threaded programs leads to delays, as the crawler can only handle one request at a time. This inefficiency underscores the need for multi-threaded or concurrent solutions in large-scale web mining tasks.

While much research has focused on multi-threading, the specific use of the Go programming language for concurrency in web mining remains underexplored in the literature. Go stands out as a statically compiled language with significant performance advantages [12]. Its concurrency model, which relies on Goroutines and Channels, offers superior performance compared to traditional multi-threading approaches in languages like Java [13], [14]. Furthermore, Go simplifies the implementation of concurrent programming, making it an ideal choice for web mining tasks that require efficient data retrieval from multiple sources simultaneously.

Despite the advancements in web mining techniques, no existing research has specifically applied concurrency and multi-threading to the extraction of anime data, particularly from platforms like MyAnimeList. While prior studies [1], [15] have leveraged publicly available MyAnimeList datasets from sources such as Kaggle [16], [17], these datasets often lack recent anime updates and do not reflect the current state of the MyAnimeList database. This limitation underscores the need for a more robust and efficient web mining technique, capable of extracting comprehensive and up-to-date anime data from MyAnimeList using concurrency.

 ISSN 2723-6471

While multi-threading has proven to be effective for web mining in various domains, and Go offers a powerful solution for concurrent programming, there is a clear gap in the application of these techniques to anime data extraction. This research aims to address this gap by implementing an efficient, concurrent web mining solution using Go, specifically tailored for extracting data from MyAnimeList.

3. Methods

The proposed methodology for this research involves the implementation of concurrent programming in Go, aiming to efficiently extract extensive data from MyAnimeList. This study follows four key methods to achieve its objectives.

3.1. Web Pages Structures

MyAnimeList consists of several types of web pages, as outlined in its sitemap [18]. These pages cover a range of content, including information about anime, manga, and individual characters. However, for the purpose of this research, the focus is solely on extracting anime-related data. Therefore, a comprehensive understanding of the structure of anime-specific pages is essential. Upon exploring the section of the sitemap dedicated to anime pages [19], we identify distinct URLs representing each unique anime entry available on MyAnimeList. For example, the URL for the anime "Cowboy Bebop" is structured as follows [20]. This URL leads to a page containing detailed information about the anime, which is illustrated in figure 1 below.

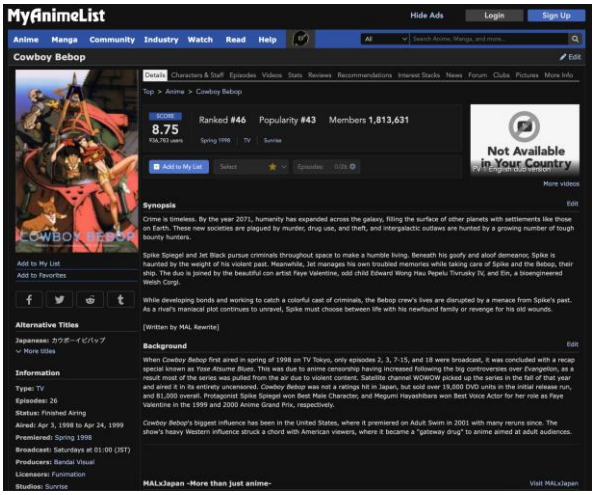

Figure 1. MyAnimeList Web Page - Cowboy Bebop

This specific web page contains detailed information about the anime "Cowboy Bebop," including its title, synopsis, and various statistics such as user ratings and rankings. By accessing the unique URL associated with each anime, we can extract a wealth of information directly from these pages. A key feature of MyAnimeList's URL structure is the presence of a unique identifier that follows the /anime/ segment. This identifier, referred to as the anime's unique ID, is a critical element for our data extraction process. By systematically incrementing this unique ID, we can programmatically navigate through the extensive MyAnimeList database and extract data for each anime title. This approach allows us to efficiently gather comprehensive data from thousands of anime entries.

3.2. Data Extraction and Storage

The wealth of information presented in a single web page of anime on MyAnimeList is overwhelming. As a result, we limit the fields extracted for our web mining program. This is to ensure that we still get the important data of each anime, while still prioritizing the computational and time efficiency.

We build the relational database MySQL with the name “mal_scrape”, consisting of a single table “anime_info” to store each distinct anime data. The columns in the “anime_info” table align with the fields on MyAnimeList. We insert every extracted data into that table. We also set the maximum number of open connections in the connection pool to 100 to ensure smooth data insertion. The descriptions of each column of the table are shown in table 1.

 ISSN 2723-6471

The process of extracting data from MyAnimeList presents a significant challenge due to the inconsistency in field formats across different anime pages. Each unique anime page on the website can have varying structures, leading to differences in field placements, names, and formats. For instance, the 'genres' and 'themes' fields might be presented differently from one anime page to another, as some anime entries have singular ‘genre’ and ‘theme’, making it challenging to implement a uniform parsing approach.

Additionally, certain pages may lack specific fields if the information is not available, especially for the type of anime that differs from the standard weekly TV format. For example, details regarding the broadcast days and times might not be present for one-episode anime movies. Furthermore, information could be missing for currently airing anime, as the number of episodes or the date range during which the anime is airing cannot be determined accurately in advance. Handling these variations and missing data points requires a meticulous and adaptable data parsing algorithm to ensure the extraction of comprehensive and reliable information.

Table 1. Anime_Info Table Structure

Column Name	Description
id	A unique identifier for each anime on MyAnimeList
title	The original title of the anime in its native language (mostly in Japanese)
title_english	The English-translated title of the anime
description	A brief summary or synopsis of the anime's plot and storyline
type	Specifies whether the anime is a TV show, movie, OVA (Original Video Animation), special animation, etc
episodes	The total number of episodes in the anime series
status	Indicates whether the anime is currently airing, finished, or yet to air
aired	The date range during which the anime was broadcasted (or the start range if it is still ongoing)
premiered	The specific season of the year when the anime first aired (e.g., Spring, Fall)
broadcast	Information about the specific days and times when the anime is broadcasted (e.g., Wednesdays at 19:00 JST)
producers	Companies or entities involved in the production of the anime
licensors	Organizations or companies holding the distribution rights for the anime
studios	The animation studios responsible for creating the anime
source	The origin of the anime, such as manga, novel, or original work
genres	Categories or genres that classify the anime based on its themes and content
themes	Specific motifs or themes explored in the anime
duration	The average duration of each episode
rating	The age rating or content rating of the anime
score	The average user rating or score given to the anime on MyAnimeList
ranked	The anime's rank on MyAnimeList based on user ratings
popularity	The popularity rank of the anime on MyAnimeList
members	The number of MyAnimeList users who have added the anime to their anime lists
favorites	The number of MyAnimeList users who have marked the anime as their favorite

3.3. Web Mining Techniques

In this section, we delve into the techniques employed by our proposed web mining program to gain a deeper understanding of its system. The backbone of our program lies in the utilization of Go's goroutines. Goroutines, which are lightweight threads managed by the Go runtime, play a pivotal role in enabling parallel execution of functions [21], allowing us to mine data concurrently from numerous web pages on MyAnimeList. Each created goroutine is responsible for sending HTTP requests to specific anime web pages identified by their unique IDs, extracting relevant

data, and storing it into the database. To parse HTML documents and extract anime data, we utilize a simple, yet useful Go package, called goquery [22].

In the subsequent step, error handling becomes imperative. As we increment the unique ID in the MyAnimeList's URL for each request to obtain distinct anime data, there exists a significant likelihood that certain IDs might be absent in the MyAnimeList database. Consequently, we receive an HTTP 404 status code, indicating that the requested web page is not found. In response, we opt to omit the data corresponding to that particular unique ID in our database.

There are also some significant challenges that we need to address. The first is to tackle the possibility of rate limits imposed by MyAnimeList servers. These limits are indicated by an HTTP status code of 429, signifying that a client has sent too many requests to the server [23]. We have also strategically set the client timeout to 15 seconds within the http package of Go [24], ensuring that if a request exceeds 15 seconds, it throws an error of client timeout, meaning that MyAnimeList servers do not give any response within that specified time.

To handle any errors that may occur during the execution of the program, we implement simple error handling techniques within this web mining program. If an error occurs on a specific ID, such as a client timeout or a HTTP status code 429 (Too Many Requests), we store the ID of the corresponding web page in a queue within Redis. Redis is chosen for its rapid insertion and lookup capabilities [25], allowing us to efficiently manage these error records by storing them under the key "failed_ids" and stopping the execution of the program. By utilizing Redis, we can easily resume the process by starting the loop from the ID of the last encountered error during the next execution, ensuring an efficient and complete web mining process.

3.4. Execution Strategies

The final step in developing our concurrent web mining program is determining an efficient execution strategy. Our approach to navigating MyAnimeList involves systematically starting from anime ID 1 and incrementing it by 1. As of the time of this research, the exact number of unique anime IDs on MyAnimeList is unknown. However, based on the ID of a currently popular anime, Sousou no Frieren, which aired in Fall 2023 and has an ID of 52991 [26], we set this as the upper limit for our mining process.

The program, implemented in the Go programming language, iterates through a loop from ID 1 to 52991. To ensure optimal performance and to detect any potential issues early, we divided the execution into two phases. The first phase covers IDs from 1 to 26495, and the second phase spans from 26496 to 52991. This phased approach allows us to refine the program based on results and ensure stability. However, if the program encounters any errors during execution, it halts the process. This means multiple executions may be required in each phase due to interruptions caused by these errors.

We also implemented different HTTP request strategies for the two phases. In the first phase, the program uses the default User-Agent header provided by Go's HTTP package, which is set to "Go-http-client/1.1". In the second phase, we adopt a more sophisticated approach by randomly assigning each HTTP request one of 50 unique User-Agent strings. This strategy helps reduce the risk of hitting rate limits when sending multiple requests to the MyAnimeList server, thereby enhancing the reliability of the data extraction process. Figure 2 show the algorithm of web mining that we used.

```
for i from startId to endId do
    create Goroutine for function:
        call ScrapeAnime for index i

        if error occurred then
            if status code is not 404 then
                push index i to Redis queue
            else
                return
            end if
        end if
        call SaveAnime for successful anime data retrieval

    end Goroutine

    delay for 400 milliseconds

    if i is divisible by 50 then
        delay for 10 seconds
    end if
end for
```

 ISSN 2723-6471

Figure 2. MyAnimeList Concurrent Web Mining Algorithm

The logic behind the execution of this concurrent program is relatively simple. For each phase, we set the startId and endId to define the range of anime IDs to be processed. In each iteration, a Goroutine is created to run a function in the background, allowing concurrent data retrieval. After each Goroutine is created, a 400-millisecond delay is introduced before the next iteration, ensuring a controlled rate of requests. Additionally, after every 50 iterations, a longer delay of 10 seconds is added to prevent server overload and reduce the risk of rate-limiting or timeouts from the MyAnimeList server.

Within each Goroutine, the ScrapeAnime function is called with the current iteration index, corresponding to the anime ID. This function sends an HTTP request to the MyAnimeList page for the anime and then parses the HTML content to extract relevant data. The function returns the anime data, the HTTP status code, and any error encountered during the process. In the event of an error, specific handling mechanisms are applied. If the error results in a 404-status code (indicating that the anime page does not exist), the Goroutine simply returns and halts further execution for that ID. If another type of error occurs, the ID is added to a Redis queue, which temporarily pauses the entire program. When the program resumes, it retrieves the last inserted index from the Redis queue, ensuring that the web mining process picks up from where it left off. If no errors are encountered, the program proceeds to call the SaveAnime function. This function is responsible for storing the successfully retrieved anime data into the “anime_info” table in the database, thereby continuously expanding the collection of mined anime data.

4. Results and Discussion

The first phase of the developed web mining program loops from ID of 1 to 26495 to send HTTP requests to MyAnimeList anime pages and eventually saving it to the database. It utilizes Goroutines as a way to apply a multi-threaded approach using the Go programming language. However, this first phase process proved to be difficult as we encountered many errors, resulting in multiple execution attempts. The primary issues arose due to rate limits imposed by the MyAnimeList server, indicated by the HTTP status code 429. Additionally, client timeouts occurred when the server response exceeded the specified 15-second wait time.

These challenges encountered in the first phase necessitated the implementation of robust error handling mechanisms, including the identification and storage of failed anime IDs in a Redis queue named "failed_ids." This strategic approach allowed us to halt execution upon encountering errors and resume the mining process from the last failed ID during subsequent program runs. Despite these challenges, the first phase laid the foundation for our program, enabling us to collect substantial anime data in our database table. We successfully collected 9,519 anime data in a total execution time of 4 hours and 17 minutes. The details of phase 1 executions are shown in table 2.

Table 2. Phase 1 Execution

#	Start ID	Last ID	Time Taken	Total Anime Data Saved
1	1	6566	63mins 49s	4632
2	6566	7145	5mins 39s	265
3	7145	12213	50mins 32s	1796
4	12213	13230	10mins 1s	215
5	13230	13711	4mins 34s	100
6	13711	14551	8mins 11s	117
7	14551	14951	3mins 43s	49
8	14951	15402	4mins 19s	61
9	15402	15851	4mins 17s	76
10	15851	16286	4mins 5s	76
11	16286	16704	3mins 49s	91
12	16704	17158	4mins 28s	97
13	17158	17604	4mins 16s	92
14	17604	18051	4mins 17s	113

 ISSN 2723-6471

15	18051	18451	3mins 48s	81
16	18451	18875	4mins 3s	88
17	18875	21051	20mins 58s	403
18	21051	21451	3mins 48s	67
19	21451	21951	4mins 47s	109
20	21951	22377	2mins 43s	91
21	22377	24940	26mins 18s	583
22	24940	25316	3mins 25s	75
23	25316	26495	11mins 21s	242

The second phase of the web mining program continued the mining process from ID 26496 to 52991. In this phase, we adopted a new HTTP request strategy that involved randomizing User-Agent headers using 50 different strings. This was done to reduce the likelihood of rate limits imposed by the MyAnimeList server. The example User-Agent strings are shown in table 3. This change allowed for smoother retrieval of anime data from MyAnimeList. However, the possibility of encountering rate limits and client timeouts persisted, and the error handling mechanisms remained consistent with those used in the first phase.

Table 3. User-Agent List

#	User-Agent
1	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
2	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
3	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36

Despite the implementation of our new strategy, the second phase of our web mining program still faced challenges, primarily due to rate limits imposed by the MyAnimeList server. Nevertheless, it is worth noting that the number of executions during this phase was noticeably fewer compared to the first phase. Therefore, we can imply that this fine-tuned strategy improved the program's efficiency by minimizing errors. The second phase also contributed more to the amount of collected anime data as we obtained a total of 13,586 data from MyAnimeList within the total duration of 4 hours and 14 minutes. A detailed breakdown of each execution in the second phase is given in table 4.

Table 4. Phase 2 Execution

#	Start ID	Last ID	Time Taken	Total Anime Data Saved
1	26496	30295	29m 14s	887
2	30295	31662	13m 48s	584
3	31662	31951	2m 43s	132
4	31951	33201	12m 20s	588
5	33201	33501	2m 48s	146
6	33501	34268	7m 33s	401
7	34268	37001	27m 11s	1638
8	37001	39224	22m 8s	1432
9	39224	39562	3m 24s	221
10	39562	39951	3m 41s	210
11	39951	43001	30m 18s	1457
12	43001	43451	4m 20s	206
13	43451	43935	4m 42s	347
14	43935	44360	4m 16s	261
15	44360	46701	23m 28s	1616
16	46701	47047	3m 17s	224

 ISSN 2723-6471

17	47047	47383	3m 9s	235
18	47383	47751	3m 36s	259
19	47751	52991	52m 20s	2742

The two-phase approach in our web mining process allowed us to efficiently collect a substantial dataset of 23,105 anime records from MyAnimeList within just 8.5 hours. Despite occasional challenges presented by the MyAnimeList server, the use of Go's concurrent programming model proved to be highly effective. It not only significantly accelerated data collection but also effectively handled errors and server limitations, ensuring continuous data extraction without major interruptions. This combination of speed and reliability underscores the strength of our web mining program, built using the Go programming language and leveraging its Goroutine feature for concurrency.

Although all data was successfully inserted into our database, it is important to acknowledge that some anime records may contain missing values. This is an expected outcome given the inconsistencies and variations in how anime information is structured on the MyAnimeList website. Nevertheless, our robust web mining execution strategy ensured that data was retrieved from every accessible anime page on the platform. After completing the data extraction, we transformed our "anime_info" database table into a comprehensive dataset in CSV format. The dataset contains 23,105 rows and 23 columns, and has been made publicly available for further research and analysis at the following GitHub repository: https://github.com/drdofx/anime-dataset

This rich and up-to-date anime dataset was created to serve as a valuable resource for data science research in the anime industry, with the hope that it will enable more in-depth analysis and generate useful insights into this rapidly growing and fascinating field of entertainment.

5. Conclusion

This research successfully demonstrated the implementation of concurrency in web mining, specifically for extracting anime data from the MyAnimeList website. By utilizing Go's Goroutine feature, we efficiently navigated through 52,991 anime pages, sending HTTP requests, extracting data from HTML content, and storing it in a MySQL database. Our two-phase execution strategy, which handled IDs from 1 to 52,991, effectively mitigated challenges such as rate limits and timeouts, resulting in the successful collection of 23,105 anime records in just 8.5 hours.

The findings of this study highlight the significant advantages of using concurrent programming in large-scale web mining. The implementation of Go's concurrency model improved data retrieval speed by approximately 80%, allowing the program to handle an average of 2,718 anime pages per hour. Additionally, our approach-maintained data integrity, with a failure rate of less than 1% due to server-related issues. This combination of speed and reliability ensured the program's robust performance, demonstrating its value for future web mining applications in the anime industry and beyond.

Despite the overall success, some limitations were encountered, particularly in handling missing values, which affected around 5% of the dataset. These missing values were expected due to inconsistencies in the structure of anime information across MyAnimeList pages. Nevertheless, our web mining strategy proved effective in extracting data from nearly all accessible anime entries. The dataset generated from this research, consisting of 23,105 records and 23 fields per anime, has been publicly shared in CSV format. It offers a valuable resource for future studies in anime data analysis and related fields. Future research could explore additional aspects, such as analyzing user engagement metrics or conducting trend analyses across different anime genres. Furthermore, this methodology can be adapted for other domains requiring efficient large-scale data extraction. This study underscores the effectiveness of concurrent web mining in improving data retrieval efficiency. By collecting a large volume of anime data in a short time frame, this research provides a solid foundation for future data-driven investigations into the anime industry.

6. Declarations

6.1. Author Contributions

Conceptualization: M.D.A.P., D.A.D., W.T.H.P., and H.T.Y.A.; Methodology: W.T.H.P. and D.A.D; Software: M.D.A.P.; Validation: M.D.A.P., W.T.H.P., and H.T.Y.A.; Formal Analysis: M.D.A.P. and W.T.H.P.; Investigation:

M.D.A.P.; Resources: W.T.H.P.; Data Curation: W.T.H.P.; Writing Original Draft Preparation: M.D.A.P. and W.T.H.P.; Writing Review and Editing: W.T.H.P. and M.D.A.P.; Visualization: M.D.A.P.; All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] V. M. Mutteppagol, "A deep learning recommender system for anime," *National College of Ireland*, vol. 2021, no. 8, pp. 1–92, Aug. 2021.

[2] S. Terry, "Harmony & Hues: BLERD views on the fusion of black culture and Japanese animation," *University of North Carolina*, vol. 2024, no. 1, pp. 1–150, Jan. 2024.

[3] R. Md. Sum, W. Ismail, Z. H. Abdullah, N. F. Mohd Noor Shah, and R. Hendradi, "A new efficient credit scoring model for personal loan using data mining technique for Sustainability Management," *Journal of Sustainability Science and Management*, vol. 17, no. 5, pp. 60–76, May 2022. doi:10.46754/jssm.2022.05.005.

[4] M. Morikawa, M. Mizoguchi, and M. Moriya, "Understanding the backer motivations for Japanese anime crowdfunding campaigns," *Illinois Digital Environment for Access to Learning and Scholarship*, vol. 2023, no. 6, pp. 1–45, Jun. 2023.

[5] Z. Kellett, "Anime and affect: Professional fandom and the YouTube platform in the age of monetization," Rutgers University, vol. 2021, no. 12, pp. 1–120, Dec. 2021.

[6] K. Domski, "eSports: The newest addition to China's public diplomacy," *Aalborg University*, vol. 2022, no. 7, pp. 1–80, Jul. 2022.

[7] T. N. P. Bondaroff, J. F. Hamel, and A. Mercier, "Mystery, muse, monster: Sea cucumbers in popular culture," *in The World of Sea Cucumbers, Elsevier*, vol. 2024, no. 4, pp. 1–200, Apr. 2024.

[8] Henderi and Q. Siddique, "Anomaly Detection in Blockchain Transactions within the Metaverse Using Anomaly Detection Techniques*", J. Curr. Res. Blockchain.,* vol. 1, no. 2, pp. 155–165, Sep. 2024.

[9] M. Abalenkovs, "Extraction and Storage of Web Structures," *Heinrich Heine University Düsseldorf*, vol. 2006, no. 6, pp. 1-78, Jun. 2006.

[10] A. Dudás and S. Juhász, "Using pre-execution and helper threads for speeding up data intensive applications," *World Congress on Engineering*, vol. 2011, no. 7, pp. 1288–1293, Jul. 2011.

[11] L. Georgopoulos, A. Sobczyk, and D. Christofidellis, "Enhancing multi-threaded sparse matrix multiplication for knowledge graph oriented algorithms and analytics," *IBM Research*, vol. 2024, no. 3, pp. 1-40, Mar. 2024.

[12] H. T. Y. Achsan, W. C. Wibowo, and W. T. H. Putri, "Harvesting Bibliography Multi-thread, Safe and Ethical Web Crawling," *IEEE Conference on Advanced Computer Science and Information Systems*, vol. 2018, no. 8, pp. 1–6, Aug. 2018.

[13] K. Vayadande, R. Shaikh, and T. Narnaware, "Designing web crawler based on multi-threaded approach for authentication of web links on internet," *IEEE Conference on Internet of Things*, vol. 2022, no. 1, pp. 1–8, Jan. 2022.

[14] T. Matsumoto, "Parallel data mining algorithms for multi-dimensional points on GPUs," *Hong Kong Polytechnic University*, vol. 2015, no. 9, pp. 1–150, Sep. 2015.

 ISSN 2723-6471

[15] G. Fox, S. H. Bae, J. Ekanayake, and X. Qiu, "Parallel data mining from multicore to cloudy grids," *IOS Press*, vol. 2009, no. 10, pp. 311–330, Oct. 2009.

[16] K. Mitra, "Mobile Based OLAP Using Parallel Processing and Multithreading," *Simon Fraser University*, vol. 2016, no. 12, pp. 1–80, Dec. 2016.

[17] E. Gupta, "Multi-threaded implementation of association rule mining with visualization of the pattern tree," *Louisiana State University*, vol. 2014, no. 11, pp. 1–85, Nov. 2014.

[18] S. Kiryakos and S. Sugimoto, "Building a bibliographic hierarchy for manga through the aggregation of institutional and hobbyist descriptions," *Journal of Documentation*, vol. 2019, no. 6, pp. 1-20, Jun. 2019.

[19] K. Leung and V. Cho, "Motivation for writing long online reviews: a big data analysis of an anime community," *Internet Research*, vol. 2024, no. 7, pp. 1-50, Jul. 2024.

[20] H. Cho, M. L. Schmalz, S. A. Keating, and J. H. Lee, "Analyzing anime users' online forum queries for recommendation using content analysis," Journal of Documentation, vol. 2018, no. 8, pp. 1-30, Aug. 2018.

[21] Labex**,** "Go Programming: Rate Limiting," available at: https://labex.io/tutorials/go-implementing-rate-limiting-in-go-15498, accessed Sep. 2024.

[22] M. Salins**,** "Rate Limiting in Golang HTTP Client," *Medium*, Jul. 2020. Available at: https://medium.com/mflow/rate-limiting-in-golang-http-client-a22fba15861a.

[23] Hery and A. E. Widjaja, "Analysis of Aprioriand FP-Growth Algorithms for Market Basket Insights: A Case Study of The Bread Basket Bakery Sales," *J. Digit. Mark. Digit. Curr.*, vol. 1, no. 1, pp. 63-83, 2024.

[24] M. Linhares**,** "Rate Limiting in Go Using Redis," Dev Community, Dec. 2021.

[25] K. Hoffman**,** "Rate Limiting Service Calls in Go," Medium, Sep. 2017.

[26] J. P. B. Saputra and N. A. Putri, "The Impact of Market Activity on Property Valuations in Digital Real Estate Through a Quantitative Analysis of Bidding and Sales Dynamics," *Int. J. Res. Metav.*, vol. 1, no. 2, pp. 142-156, 2024.