Finding One Piece: Episode Search System

Hugo Castro
up202403131@fe.up.pt
FEUP
Porto, Portugal

Marcos Costa
up202108869@fe.up.pt
FEUP
Porto, Portugal

Rodrigo Moucho
up202108855@fe.up.pt
FEUP
Porto, Portugal

ABSTRACT

This project aims to develop a search engine for the anime series One Piece, a long-running and widely popular work in the anime industry about a crew of pirates and their adventures. The project involves the collection of data from two primary sources: the One Piece Fandom Wiki and MyAnimeList. The data processing pipeline was done using Selenium with Python for web scraping, followed by data cleaning and analysis using the pandas library. The search functionality is being developed in Apache SOLR[6]. The analysis helped to understand the data on a deeper level including character-frequency and common themes. Many models were developed to structure and analyse the collected data. These models were then evaluated based on the results they produced for an in-depth statistical analysis. For the best model the MAP was 0.53 and the P@20 was 0.73. A GUI was also developed for easier and more intuitive access to the search engine.

CCS CONCEPTS

• **Information Systems** → **Information Retrieval**; **Data Mining**.

KEYWORDS

One Piece, Episodes, Dataset, Data Pipeline, Data Cleaning, Data Processing, Data Analysis, Search Engine, Information Retrieval, Web Scraping

ACM Reference Format:
Hugo Castro, Marcos Costa, and Rodrigo Moucho. 2018. Finding One Piece: Episode Search System. In *Proceedings of (G15).* ACM, New York, NY, USA, 11 pages.

1 INTRODUCTION

This milestone focuses on the development of a data extraction pipeline designed to collect detailed information from the One Piece Fandom Wiki, specifically targeting episodes of the anime One Piece and data from MyAnimeList, specifically, the user ratings of the said episodes.

The motivation for this work originates from the growing demand and interest for japanese media and entertainment.

G15, October-14, 2024, Porto,Portugal

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

By focusing on One Piece, one of the biggest anime series in the world, we try to address this need for fans who engage with the series.

Additionally, since this particular series has such a large amount of information due to the high number of episodes and characters, future users will benefit from improved accessibility in exploring One Piece data.

Our goal is to use the extracted episode data, that includes the episode title, long and short summary, air date, featured character, featured Devil Fruits and respective episode rating to develop a search engine.

This document highlights both the retrieval and processing of the chosen data, and is structured in several major sections.

In Data Sources we identify the sources of the data and briefly characterize the dataset.

The following section is Collection and Preparation where we describe the developed pipeline and highlight the collection and processing operations as well as the datasets' conceptual data model.

In the Characterization section we study the scraped dataset with a variety of exploratory and descriptive statistics and analysis.

Lastly Prospective Search Tasks and Conclusions outlines possible future work for the evolving project.

2 DATA SOURCES

The primary data source for this project is the One Piece section of the Fandom Wiki [3] and MyAnimeList (MAL) [2].

2.1 Identification

We are utilizing the Fandom Wiki, which operates under a free-to-use model but has its own licenses for user-generated content.

This specific platform falls under the Creative Commons Attribution-ShareAlike (CC BY-SA) license, which allows for redistribution and modification as long as appropriate credit is given and derivative works are shared under the same conditions.

However, MyAnimeList maintains complete copyright over its material, and because its data is publicly available but not openly licensed for reuse, it is utilized for academic purposes in accordance with fair use principles

2.2 Characteristics

The Fandom Wiki is a cooperative platform where fans contribute detailed information about a wide array of media, including, but not limited to, anime series like One Piece. For this project, we focus on scraping key fields such as episode titles, summaries, air dates, and featured characters.

The long episode summary is also valuable to us since it contains references to Devil Fruits—powers in the One Piece world. Our code is designed to identify words that follow the pattern " *blank blank*no Mi" to automatically detect the presence of these foods.

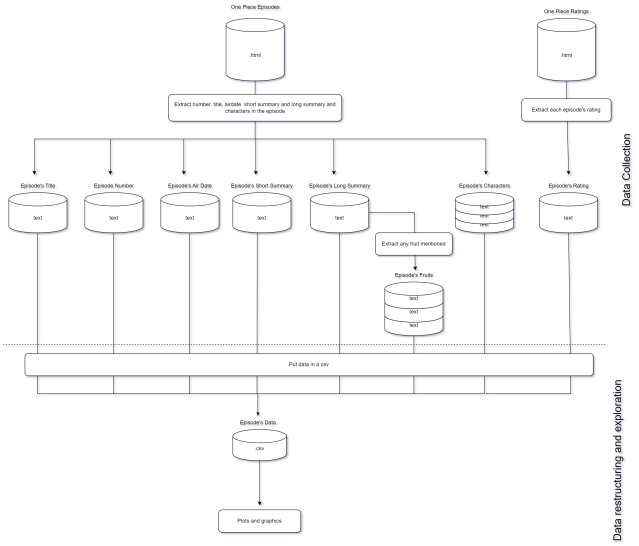

Figure 1: Pipeline for data collection and preparation

MyAnimeList is an anime database, widely used by all kinds of anime fans to track their progression on their favorite animes, it also possesses the feature to rate individual episodes, which will be of interest to us.

The dataset has a total of 1120 rows (each representing an episode) and every column is text data, excluding the rating one, which is numerical.

3 COLLECTION AND PREPARATION

In this section, we expose the method used to collect the data and prepare it for usage.

3.1 Pipeline Description and Diagram

Here we present the steps executed to obtain the data and process it to make it ready to be analyzed.

The goal of this pipeline is to retrieve the data from an exterior source, mainly two websites, One Piece Fandom (for episode details) and MyAnimeList (for episode ratings), and convert it into a database so that it is easier to manipulate and examine.

The first step of the pipeline is to extract the information, which we do using Selenium, a package used to automate web browser interaction from Python[5].

The script automatically navigates through each episode's web page within the One Piece Fandom website and retrieves relevant information such as the episode number, title, air date, summaries, characters, and magic fruits.

It also scrapes episode ratings from MyAnimeList, associating them with the corresponding episode.

In Figure 1 is a diagram that visualizes the pipeline and the steps involved in data collection and preparation.

3.2 Collection Operations

The collection operations performed by the script focus on navigating the target websites and extracting the required information from elements located using the absolute path, XPath.

Other operations include accepting cookies, and locating web elements whose XPath isn't consistent across different pages. Each page is loaded and it's relevant content is parsed before moving to the next episode.

3.3 Processing Operations

The processing operations include obtaining the mentions of magic fruits in the episodes and storing the results in a final file, as well as cleaning the data.

The script extracts any mention of magic fruit within the long summary of each episode, obtained in the previous step, using regex patterns.

The extracted data is then formatted and stored in a CSV file.

The ratings collected from the MyAnimeList are merged with episode details, thus becoming an attribute of the episode class.

The data was analyzed for potential null values, and we identified the following number of episodes with missing details:

- **Short Summary Missing**: 1
- **Long Summary Missing**: 54
- **Rating Missing**: 4

Upon investigation, we discovered that the missing details were due to incomplete information on the source website. Therefore, we decided to retain these episodes despite the null attributes.

Additionally, we considered normalization but found that it wasn't necessary given that each data attribute came from one source (e.g. ratings from MyAnimeList or short summaries from the Wiki Fandom), and because of that all the data had a consistent format.

In the end, we get structured data ready for analysis.

3.4 Conceptual Data Model

The conceptual data model includes the classes, attributes and connections between classes. The conceptual data model of our data is shown in the Figure 2.

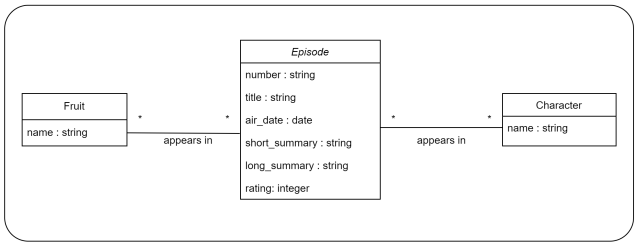

Figure 2: Conceptual Data Model

3.5 Structured and Reproducible Pipeline

The code is consolidated into a single script that automates all steps of data collection and processing described above. It's designed

for reusability, allowing the same process to be rerun when new episodes are released.

The script is documented to facilitate maintenance and future modifications, making it adaptable to different data requirements or changes in the websites.

4 CHARACTERIZATION

In this section, we'll dive into the characterization of our collection, exploring different facets like the structure of our documents and conducting some exploratory data analysis to look at various variables.

4.1 Character Statistics

Analysing Figure 5 it can be observed the characters that appear in most episodes.

As expected, the main character of the show Luffy is first, appearing in 800 out of 1120 episodes. He is followed by the rest of the member of his pirate crew, notably the second and third member, Zoro and Nami respectively.

Unlike in many other TV series, One Piece's main character does not appear in about 30% of the show's episodes.

This can be explained by the enormous world-building done by the writers that includes a vast number of characters, locations, and crews alongside a huge number of episodes.

By dedicating substantial time to these side-stories the narrative is able to continue without the need of Luffy.

4.2 Rating Statistics

As illustrated in Figure 6, the average episode ratings, grouped in ranges of 100 episodes, consistently rank between the 4.2 and 4.5 rating.

Notably, the episodes with the highest ratings are found in the ranges between episodes 300-600 and 900 to the present.

This shows that a pattern exists between the different episodes ratings, either slowly improving closer to the 4.5 mark or slowly decreasing towards the 4.2 rating with no abrupt changes happening.

4.3 Word Count Statistics

Figures 7 and 8 show the word count in both short and long summaries of the episodes. Excluding the first 300 episodes, the trends in both graphs show a similar pattern.

In more recent episodes, the summaries are shorter on average. A possible reason for this is because they are more recent and fewer people have had the time to expand them.

Additionally, the episodes between 500 and 700 feature longer summaries, which may be due to the fact that this period had fewer filler episodes (episodes that don't contribute to the plot/ stand-alone episodes), allowing for a bigger amount of significant information.

4.4 Text Analysis

The word cloud, shown in Figure 4, taken from the short and long summaries of the episodes provides information on the most common words and entities throughout the collection.

Unsurprisingly, the main characters of the series are very prominent in the image as well as specific entities and verbs very related to the theme of the show such as: crew, Straw Hat, Marine and ship.

To confirm our findings and deepen our knowledge about the data, we did some keyword extraction and the top-20 results were very similar to those obtained in the Word Cloud in Figure 4. We can divide these word in three main topics: Key Characters, Key Concepts and Thematic Elements.

The key characters found were chopper, doflamingo, franky, law, luffy, nami, robin, sanji, usopp and zoro. They all contribute greatly to the narrative plot of the series.

The key concepts are terms that are commonly associated with the One Piece narrative. The words found: crew, pirates, ship and straw hats, perfectly describe the setting and backdrop of the whole show.

Adding to the key concepts, the thematic elements give us a bigger picture of what are the actual contents of the story and what the characters are usually doing.

This topic contains the words big, says, tells and attack.

These terms are less specific but can indicate various narrative elements. For example, "big" could refer to a massive or powerful event, the words "says" and "tells" may reflect important dialogue and "attack" might imply action sequences or conflicts.

Figure 3: Word Cloud

5 PROSPECTIVE SEARCH TASKS

The finished word cloud in [Figure 4] gives us a visual representation of the most common terms in the episode summaries, highlighting major characters like Luffy, Zoro or Nami, as well as core concepts like, for example, crew and Marine.

However, thanks to the large number of episode data collected for our database, our search engine will allow the users to delve into more specific and less common details of the One Piece Universe, having the opportunity to search not only for common characters, for example, but also for very obscure ones.

There are a large quantity of query examples to use in the developed search engine.

One of those would be a user searching for episodes featuring secondary characters like Trafalgar Law or Helmeppoby using queries featuring their name.

Another one would be if they want to search for an episode featuring a certain Devil Fruit, whose naming patterns were already described above.

This search engine would also aid in the filtering of certain important events, like for example, a battle between certain characters, some key locations, or some other events that might not be the focal point of an episode but still be important to the central narrative.

This allows for the complete exploration of our dataset and could even be used to find certain specific story arcs/adventures or to filter out episodes focused on certain conflicts or interactions between characters.

Queries like "[Character 1] fights [character 2]", "[Characters 1, 2 and 3] go to [location X]" and "The [Crew name] Pirates" are examples that can help users reach information about these topics.

6 INFORMATION RETRIEVAL

In this section we will focus on developing the process of extracting episode-related information, relevant to a specific need. By reorganizing our data into structured episode documents the system enables improving search precision, accuracy and overall relevancy.

This section describes the methods used for indexing and retrieval via Apache Solr[6], which excels in high-speed processing, scalability, and the ability to handle complex queries way beyond a simple text-match.

6.1 Document definition

In this system, each document represents an episode, structured with attributes such as a title, air date, a short summary, a long summary, and a list of featured characters and Devil Fruits. These attributes are separated into individual fields, ensuring structured organization.

6.2 Indexing process

Indexing plays a vital role in making episode data searchable. Without this step, the search system would encounter inefficiencies, such as longer response times and an inability to handle non-basic queries.

Solr processes indexing through a combination of Tokenizers and Filters. Tokenizers split text into smaller tokens, based on custom rules and filters modify or enhance tokens.

Every field besides "rating" was indexed, since it is not expected to search for an episode by their specific rating.

For the indexed fields two new data types were created.

Field **comma-delimited**:

- **PatternTokenizerFactory**: Splits text into tokens wherever a comma appears
- **LowerCaseFilterFactory**: Converts all tokens to lowercase to make searches case-insensitive
- **WordDelimiterGraphFilterFactory**: Splits tokens into subwords to handle complex structures like underscores
- **EdgeNGramFilterFactory**: Generate sub-sequences of tokens for auto-completion and partial matching

Field **my-text**:

- **StandartTokenizerFactory**: Breaks text into tokens based on word boundaries
- **ASCIIFoldingFilterFactory**: Converts accented characters to their ASCII equivalents
- **LowerCaseFilterFactory**: Converts all tokens to lowercase to make searches case-insensitive
- **SynonymGraphFilterFactory**: Adds synonyms for tokens based on a synonym file
- **StopFilterFactory**: Removes common stop words
- **EnglishMinimalStemFilterFactory**: performs light stemming, reducing words to simpler forms
- **EnglishPossessiveFilterFactory**: Removes suffixes
- **EdgeNGramFilterFactory**: Generate sub-sequences of tokens for auto-completion and partial matching
- **PatternReplaceFilterFactory**: Removes exclamation and question marks

The fields title, short summary, long summary and air date used the my-text type.

Multi-value fields, such as characters and fruits, used the comma-delimited type.

This configuration optimizes the system to handle complex search scenarios while maintaining efficient processing of episode data.

Table 1: Complex Schema

Field	Type	Indexed
title	my_text	true
air_date	my_text	true
short_summary	my_text	true
long_summary	my_text	true
characters	comma_delimited	true
fruits	comma_delimited	true
rating	string	false

7 RETRIEVAL PROCESS

Two schemas were used for the retrieval process: a simple schema that uses default field types for all the attributes and a more complex schema, previously described.

The retrieval process leverages Solr's advanced querying capabilities to find and return relevant episodes based on user inputs. The same parameters were used for both schemas.

- **Query (q):** Specifies the main search terms.
- **Query Operator (q.op):** Defines logical relationships between search terms. Logical relationship AND was used, meaning all terms (or variations) in the query must be present in the matched documents .
- **Sort Filter (sort):** Sorts the results by criteria, in our case, by relevance calculated through the score.

To enhance search results, both systems also utilize Solr's edismax query parser. The following parameters were configured:

- **Query Field with Boosting (qf):** Assigns weights to prioritize specific fields during searches, in our case, title weights 6, short and long summary weight 3 and 2 respectively, and the rest of the fields all have weight equal to 1.
- **Phrase Boost Field (pf):** Prioritizes terms that match phrases within the query, assigning them higher relevance. The weights used in this parameter are the same as the one used in qf.

- **Phrase Boost Slope (ps):** Sets the maximum allowable gap between words in a query, configured as 2 in our case.
- **Query Phrase Slop (qs):** Sets the maximum number of word positions that can be swapped, also 2 in our queries.

By weighting fields through the qf parameter, the system ensures that highly relevant fields, such as titles, are prioritized over secondary ones, like summaries or character lists. The pf parameter further increases the relevance of closely matched phrases, while the ps and qs parameter maintain a balance between flexibility and contextual accuracy.

Different parameter combinations were explored in order to ensure the best balance for effective searches. Since the query had to be in natural language, term boosts and fuziness/wildcards were explored but not explicitly included.

This retrieval setup, enhanced by edismax, delivers relevancy ordered results tailored to user queries, improving navigation through episode data.

We chose to apply the edismax boosts for both schemas to ensure a fair comparison of their performance. Therefore excluding query parameters and focusing on the effectiveness of the schema design rather than highlighting the exploration done regarding influence of differential boosting strategies.

8 EVALUATION

Evaluating the system is important to understand how well our search system is working and how satisfied a possible user would be with their search results. Having this information is crucial for making changes to our system design.

A system can be evaluated in two ways:

- **The effectiveness**: the ability of the search system to find the right documents with relevant results based on the user query.
- **The efficiency**: how quickly the search system retrieves and shows the results of a query. The faster the search system is, the more efficient.

As such, a good search system must balance between effectiveness and efficiency in order to provide the best service possible to the user.

In this project, the evaluation of our search system is focused on its effectiveness. In order to do that, we came up with 5 information needs that our target user would have, and compared the obtained results against documents selected by hand and thus serving as the ground truth.

8.1 Evaluation Process

The steps to evaluate the search system for an information need can be described in the following steps:

(1) The information need must be converted into a query;
(2) The query is then processed using two different schemas: one simple schema only containing the types of the fields in the data, serving as the control group, and a schema with multiple filters aimed to improve the effectiveness of the results;
(3) To simplify the process of checking which documents are relevant or not in the dataset, we only check the documents returned by the two different schemas, limited to 40, and unify them into a single list. That is how we determine the qrels;
(4) We convert the results from the scripts and the gathered qrels into formats that are more manageable;
(5) The files obtained in the previous point are passed to the function `trec_eval`[7], which will produce performance metrics for our search system;
(6) Additionally, we also execute the Python script `plot_pr.py`, provided by the teachers, that plots the Precision-Recall Curve.

We limited the results from the queries to 40 since that produced a substantial number of results to gauge the performance of the system, without compromising the validity of the evaluation.

The steps of setting up the Solr instance with two different schemas, as well as querying those systems, converting the results and running the programs to get the metrics, are automated using a makefile adapted from the course provided tutorial code.

8.2 Performance Metrics

The performance metrics considered in our analyses are the following:

- **Mean Average Precision (MAP)**: Commonly used in the context of information retrieval, this metric computes the average of the precision values at each relevant document for a query, averaged over all queries. It provides a general idea of the search system's ability to return relevant documents in the top results, with higher MAP values indicating better overall performance in retrieving relevant documents.
- **Precision at N (p@n)**: This metric measures the proportion of the top-n documents returned by the search system that are relevant. It provides insight into how well the system ranks the most relevant documents within the first few results, with higher p@n values indicating better ranking of relevant documents.
- **Precision-Recall Curve plot**: This metric visualizes the trade-off between precision and recall across different thresholds for retrieval. The plot allows us to assess the overall performance of the search system, with the area under the curve (AUC) serving as an indicator of the system's effectiveness at balancing precision and recall.

8.3 Information Need and Respective Evaluation Results

In this subsection, we present the different information needs considered in our analyses, along with the performance metrics obtained for the simple and complex schemas and our interpretation of the data.

8.3.1 Zoro fights or battles.

Information Need: Identify episodes where Zoro is involved in a fight or battle.

The objective is to retrieve episodes where Zoro demonstrates his combat skills, whether in duels, group battles, or encounters involving notable opponents or challenges. The focus includes

episodes that highlight his swordsmanship or contributions to the Straw Hat crew's battles.

Query: "Zoro fights"

Table 2: Performance for query Q1

Metric	Basic schema	Boosted schema
AvP	0.71	0.33
P@20	0.77	0.60

Analysis of results: For this query, the basic schema outperformed the boosted schema in both MAP, P@20 and overall Recall and Precision, as seen in the table above and the graph number 11. This behavior can be explained by the boosted schema's additional filters, which made the results overly flexible and failed to return some relevant documents that were captured by the basic schema. Additionally, the boosted schema returned documents where the word "Zoro" was not present, rendering them irrelevant to the query. This issue might have arisen from the use of synonyms or fuzziness, allowing words unrelated to Zoro to match the query. The basic schema's simplicity proved more effective for this straightforward query. To address this issue, the filters could be adjusted to exclude unrelated words and characters' names when applying fuzziness or synonyms.

8.3.2 *Pirate Ships.*

Information Need: Retrieve episodes that prominently feature or reference a pirate ship.

Relevant episodes include those in which a pirate ship is featured as a significant element of the setting, narrative, or plot. This includes episodes where the pirate ship serves as a key location, plays a central role in the storyline, or acts as a driving force behind the events or conflicts depicted.

Query: "pirate ship"

Table 3: Performance for query Q2

Metric	Basic schema	Boosted schema
AvP	0.50	0.63
P@30	0.60	0.70

Analysis of results: Unlike the previous query, this one produced an expected outcome: the boosted schema outperformed the basic schema. The difference lies in the effectiveness of the filters applied in this query compared to the effectiveness of those same filters in the previous one. Here, the boosted schema excelled at narrowing down the results to episodes specifically focused on pirate ships. Its superior performance is likely due to the use of synonyms, which enabled the boosted schema to identify references to pirate ships described with different terms. In contrast, the basic schema could only retrieve documents containing the exact terms used in the query. This information need thus highlights the importance of the filters of the boosted schema.

8.3.3 *Straw Hat crew members having a reunion.*

Information Need: Locate episodes where the Straw Hat crew or any of its member has a reunion with another character.

The objective of the task is to find potential reunions between important characters that include Straw Hat crew members, which may include pivotal moments in the One Piece series.

Query: "Straw Hat crew reunion"

Table 4: Performance for query Q3

Metric	Basic schema	Boosted schema
AvP	0.25	0.49
P@30	0.20	0.43

Analysis of results: Similar to the previous query, the boosted schema outperformed the basic schema. The broader and more complex query amplified the difference between the two, highlighting the basic schema's inability to capture the context of reunions among the Straw Hat crew members. This query likely benefited from the boosted schema's enhanced filters to match words like "reunite" or synonyms such as "meet" or "join".

8.3.4 *Luffy losing a fight or battle.*

Information Need: Locate episodes where Luffy is defeated in a duel or conflict

The objective of the task is to find episodes where Luffy is defeated or loses a fight against another character.

Query: "Luffy loses battle"

Table 5: Performance for query Q4

Metric	Basic schema	Boosted schema
AvP	0.19	0.46
P@30	0.20	0.50

Analysis of results: The results for this query are consistent with those of query 3 across both evaluation metrics. For a more detailed analysis, we refer to Figure 14, where we can observe that while the basic schema returns more relevant results at lower thresholds, its performance drops significantly as the threshold increases. In contrast, the complex schema shows lower precision at the beginning but maintains a more stable performance as the threshold increases, demonstrating its ability to consistently provide relevant results at higher thresholds.

8.3.5 *Straw Hat crew moving into a new location or island.*

Information Need: Locate episodes where the Straw Hat crew travels to new and significant locations.

Relevant results include the Straw Hat crew traveling to a different island or location, specifically when it relates to plot development, making those moments particularly relevant.

Query: "Straw Hat crew new location"

Table 6: Performance for query Q5

Metric	Basic schema	Boosted schema
AvP	0.11	0.66
P@30	0.13	0.60

Analysis of results: The boosted schema significantly outperformed the basic schema for this query, with MAP increasing from 0.11 to 0.66 and P@30 from 0.13 to 0.60. As in the last queries, the filters in the complex schema played a crucial role in this improvement.

8.3.6 Overall Evaluation. To ascertain the overall performance of the system, we calculated the average of the MAP and P@30 for all queries. The results are shown in the following table:

Table 7: Overall Performance

Metric	Basic schema	Boosted schema
MAP	0.35	0.51
Mean P@30	0.38	0.54

From the data in the table, we can see both schemas were satisfactory, but the boosted schema obtained considerably better results, which was the expected outcome.

In conclusion, while the boosted schema excels by broadening the scope of results beyond exact matches or simple variations of the query, it sometimes falters when handling entities like character names. In such cases, explicit mentions of the entity are critical, and the basic schema consistently retrieves them. In contrast, the boosted schema may occasionally return results that lack the specified entity, either due to overly flexible matching or wrongly applied synonyms. This highlights a trade-off between the precision of the basic schema and the contextual flexibility of the boosted schema.

This conclusion is further evidenced by the graphs of the Precision-Recall Curves, where a pattern can be noticed: the simple schema achieves better precision at lower thresholds but experiences a significant decline as the threshold increases. Conversely, the boosted schema, while starting with lower precision, maintains a relatively high and consistent precision as the threshold increases.

9 INFORMATION RETRIEVAL IMPROVEMENTS

In order to improve the overall performance of our search engine, different ideas were explored and tested.

This section provides an overview of the improvements implemented.

9.1 Semantic Search

To enhance our search engine's capability to understand contextual nuances, we implemented semantic search, based on the methodology provided in the course tutorial.

This approach involved generating vector representations using a transformer model, which captured the semantic meaning of words within the selected fields. For our data, we applied this strategy to the title, short summary, and long summary fields, as they were deemed the most relevant for semantic search. The resulting semantic vectors were integrated into our JSON dataset.

When the user makes a query, we use the same transformer model to generate the embeddings for the query text. These embeddings, which are dense vector representations, encapsulate the semantic meaning of the input. Using K-Nearest Neighbor (KNN) search, we compare the query embeddings with the document embeddings stored in Solr to identify the most similar vectors. Since these vectors encode semantic information, the KNN search effectively retrieves documents with similar semantic meaning to the query, ensuring highly relevant and contextually accurate results.

This methodology significantly improves the relevance and precision of search results, as we will demonstrate in the M3 Evaluation section.

9.2 Schema improvements

While the current schema was satisfactory, we explored further ideas to improve it.

In our complex M2 schema, we utilized the Synonyms Graph Filter to ensure results with synonymous terms to the user query would appear. This ensured users discovered content related to their searches even if the exact word they used was not present in the results.

To build on this, we introduced custom synonyms, created specifically for the One Piece universe. These were added to our synonyms.txt file to improve the understanding of our search engine for a variety of terms and relationships within the series, in order to deliver more accurate and relevant results.

This is particularly relevant in our case because the One Piece universe is very unique, with a variety of peculiar terms and names that are not very common.

Examples of custom synonyms are:

- Different names for the same character:
  Example: The pirate Whitebeard is also known as Edward Newgate.
- Related terms for key One Piece concepts:
  Example: Synonyms were added for terms like shipwright, sailor, treasure, bounty, battle and marine.
- Linking titles or roles to specific characters:
  Example: Kaido is synonymous with Yonko (Emperor).

These improvements allow users to get more relevant results and ensures our search system aligns better with the One Piece universe.

9.3 Query parameters improvements

During our M2 Evaluation we found a higher than expected number of returned results that were not relevant. To adress this issue, we decided to refine our query parameters to make results more precise and relevant, reducing the irrelevant matches.

To achieve this, we focused on tightening the query conditions by decreasing the Query Slop **(qs)** and Phrase Slop **(ps)** parameters from the value 2 to the new value 1, reducing the occurrence of irrelevant matches

This ensures stricter matching criteria, allowing for closer alignment between query terms and returned results. This adjustment reduces the over-flexibility observed and helps to improve the relevance of the search results.

10 USER INTERFACE

In order for our users to have an easy time navigating through their desired searches, a Graphical User Interface was developed. This section will be used to describe the development of said GUI and every functionality present within it.

Since the normal Solr interface is very complicated and not user-friendly, this is a very important step to ensure our search engine is accessible to possible users.

To develop this interface, HTML and JavaScript were used in combination with Solr's API for the results and search of the queries.

To support these needs, we've created the interface which is composed of two Dynamic Pages:

- Home: where the users can type out their queries and see the results, showcased in a list below;

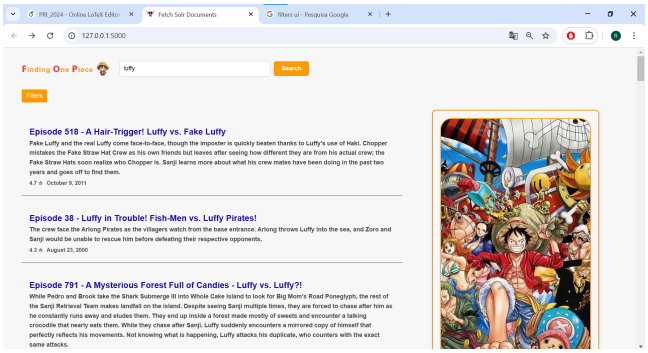

Figure 4: Home Page

- Episode: where users can see specific episodes and all related information to them.

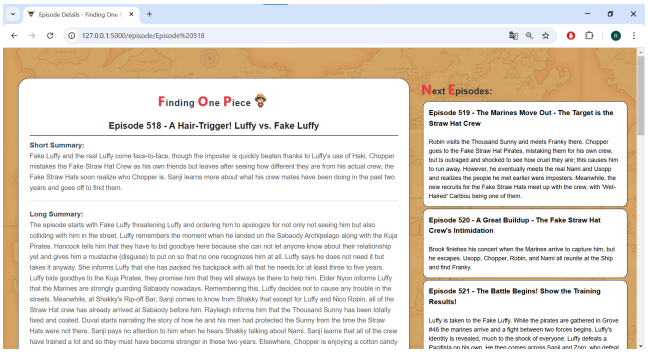

Figure 5: Episode Page

A rating filter was also developed so users can search for episodes that have a rating above a certain value, to single out the best episodes returned in a certain query.

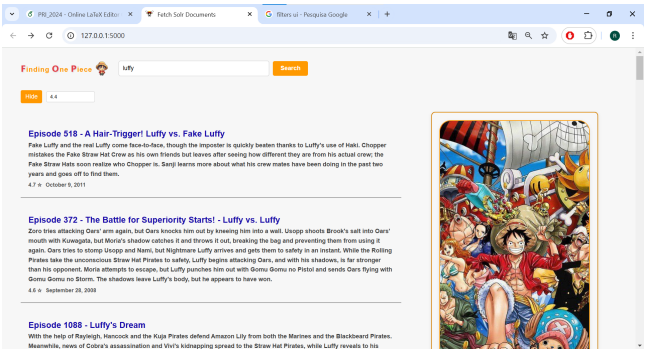

Figure 6: Home Page With Filter

All results are organized in descending order of ranking; in short, the most relevant results appear at the top, while the less relevant ones remain at the bottom.

10.1 Feedback

In order to access the quality of our GUI we gathered feedback from students at FEUP.

The general consensus was highly positive.

Students described the interface as clean and visually appealing, not overly complex and easy to use.

Overall satisfaction with the interface and it's usability was notably high.

11 M3 EVALUATION

After the changes made to the search engine, the new versions were evaluated and compared both against each other and against the M2 Boosted Schema, evaluated in Section 8. Since all the models were superior to the M2 Boosted Schema, this model is not included in the following section.

To ensure consistency and validity of the results, the evaluation process and performance metrics used were the same as those utilized in Section 8.

11.1 Evaluation Process

Three new models were developed and evaluated in order to access their performance.

- Model 1: M2 Boosted Schema incorporating the Schema improvements described in Subsection 9.2 and the Query Parameters improvements detailed in Subsection 9.3.
- Model 2: M2 Boosted Schema enhanced with Semantic Search, as explained in Subsection 9.1.
- Model 3: A Hybrid Model that combines Model 1 and Model 2 by merging their results alternately. However, to ensure that only the most relevant results from Model 1 were included, only the top three highest scored results were added. This approach guarantees that only high-quality results from Model 1 contribute to the final top-20 results.

This merge logic for Model 3 was adopted because, as demonstrated in the following subsection, Model 2 consistently outperforms Model 1, although a fraction of the most relevant results in Model 1 are not returned in Model 2.

This merge logic ensures less non-relevant results get merged into the final response while highly rated M1 model results (which have a higher probability of being relevant) still get added.

The queries utilized to test them were previously applied to the M2 Boosted Schema:

- Query 3 - "Straw Hat crew reunion"
- Query 4 - "Luffy loses battle"

The relevance of the returned results was manually assessed, with a limit of 20 results per query per model.

11.2 Evaluation Results

11.2.1 Straw Hat crew members having a reunion.

Table 8: Performance for query Q3

Metric	Model 1	Model 2	Model 3
AvP	0.27	0.47	0.54
P@20	0.40	0.65	0.70

Analysis of results: As expected the Hybrid Model (M3) outperformed the other models, closely followed by the Semantic Search model (M2).

The combination of the two models yielded slightly better results since one of the top-3 results from M1 was not included in M2.

We can observe that semantic search achieves a higher performance than the M1 Model. This highlights the importance of using context in Information Retrieval.

Table 9: Performance for query Q4

Metric	Model 1	Model 2	Model 3
AvP	0.39	0.56	0.52
P@20	0.55	0.70	0.75

11.2.2 Luffy losing a fight or battle.

Analysis of results: The results for this query show similar differences between the Hybrid Model and the other two models.

As in the last query, the Semantic Search M2 Model returns significantly better results than the M1 Model, confirming our conclusion about the importance of this type of search.

In this case, the Hybrid Search shows slightly worse AvP comparing to the M2 Model, although the P@20 is still higher.

12 FINAL SYSTEM

Table 10: Average Performance

Metric	Model 1	Model 2	Model 3
MAP	0.33	0.52	0.53
Mean P@20	0.48	0.68	0.73

The Hybrid Search Model - Model 3, calculated by merging the M1 and M2 models, is the model with the best performance, having a Mean Average Precision of 0.53 and Average Precision@20 of 0.73.

As the project reached its third milestone, we refined the system based on evaluations and feedback, arriving at a polished information retrieval solution.

13 CONCLUSIONS

In milestone one we have achieved the preparation and characterization of the datasets we selected for the project.

The chosen datasets were appropriate for the project's objective and the data sources were studied in order to assess their reliability and quality.

The processing pipeline, done using Selenium, is a reproducible and well defined automated process.

In the second milestone we defined the documents for our data and developed two different schemas to structure it.

Retrieval ideas such as boosts, slop and proximity searches were explored in order to maximize the performance of the search-engine being developed.

The created schemas were evaluated and compared statistically in order to better analyze its results and plan improvements for the future.

In the third milestone improvements were done to the search engine developed in the previous milestone.

Enhancements included query optimizations, the addition of custom synonyms, and the implementation of semantic search, which delivered substantial performance improvements, particularly semantic search.

Three different models were created and evaluated and a user-friendly GUI was developed to ensure that the search engine is accessible and easy to use for potential users.

After completing this third milestone we have successfully developed a very satisfactory One Piece search engine.

REFERENCES

[1] 2024. Mat Plot Lib. https://matplotlib.org/
[2] 2024. MyAnimeList. https://myanimelist.net/
[3] 2024. One Piece Wiki. https://onepiece.fandom.com/wiki/One_Piece_Wiki
[4] 2024. Pandas. https://pandas.pydata.org/
[5] 2024. Selenium. https://pypi.org/project/selenium/
[6] Apache. 2024. Solr. https://solr.apache.org/ Version 9.7.0.
[7] National Institute of Standards and Technology (NIST). 2024. Trec_eval: Evaluation Tool for Information Retrieval Experiments. https://github.com/usnistgov/trec_eval Version 9.0.8. Accessed: 2024-11-18.

A ANNEXES

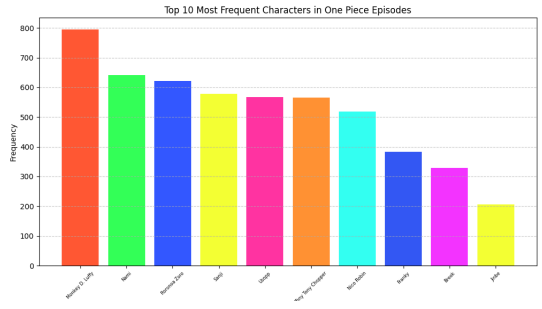

Figure 7: Top-10 Most Frequent Characters in Episodes

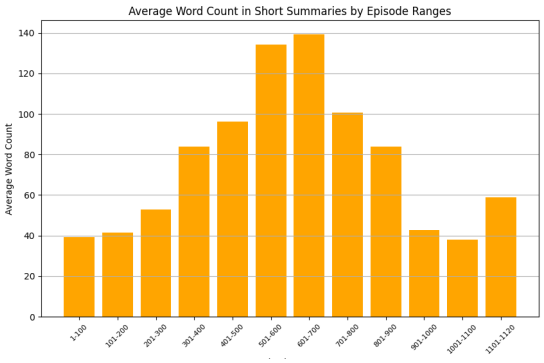

Figure 10: Average Word Count by Short Summary

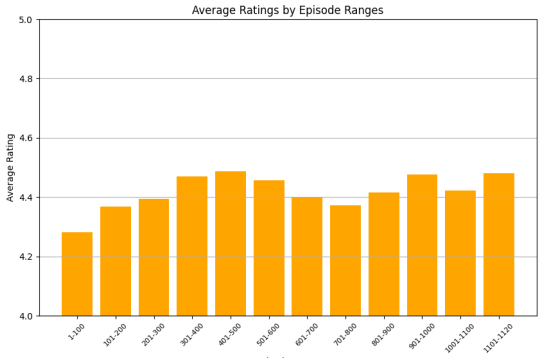

Figure 8: Average Rating by Episode

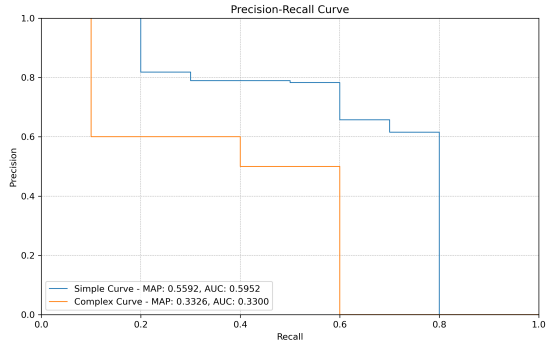

Figure 11: Precision-Recall Curve for Query 1 Boosted Schema and Simple Schema

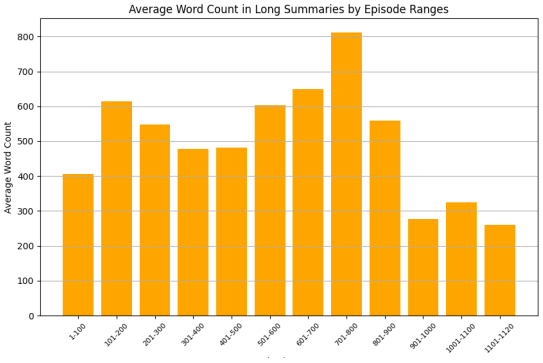

Figure 9: Average Word Count by Long Summary

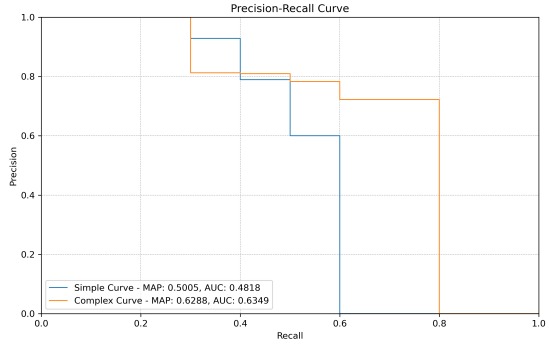

Figure 12: Precision-Recall Curve for Query 2 Boosted Schema and Simple Schema

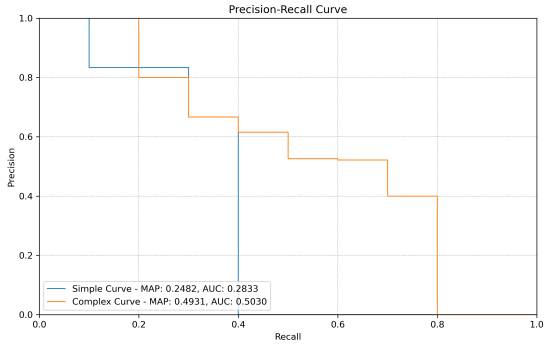

Figure 13: Precision-Recall Curve for Query 3 Boosted Schema and Simple Schema

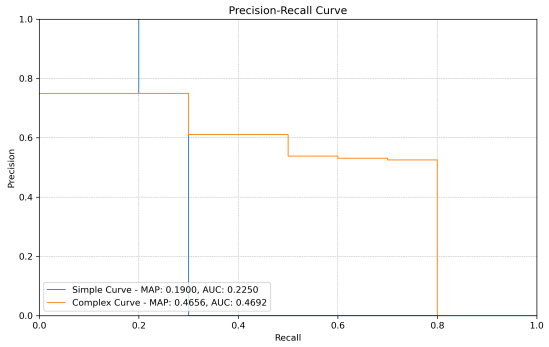

Figure 14: Precision-Recall Curve for Query 4 Boosted Schema and Simple Schema

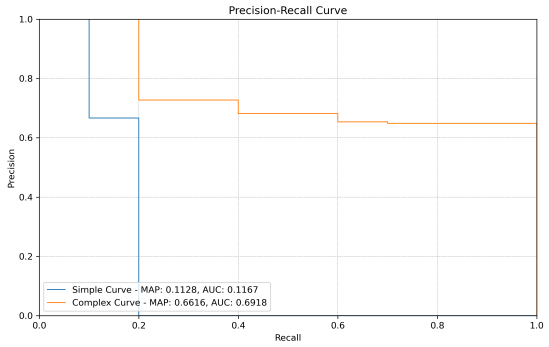

Figure 15: Precision-Recall Curve for Query 5 Boosted Schema and Simple Schema

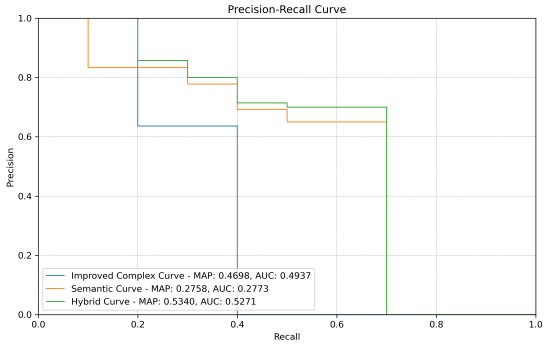

Figure 16: Precision-Recall Curve for Query 3 for 3 different schemas results.

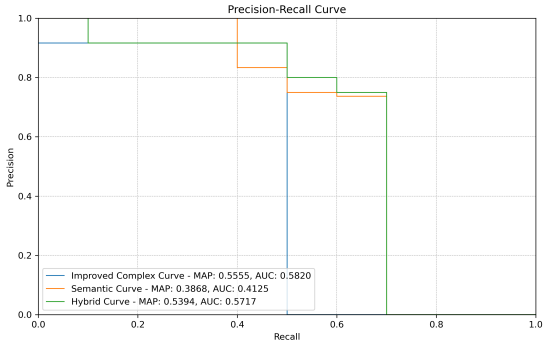

Figure 17: Precision-Recall Curve for Query 4 for 3 different schemas results.

A.1 Use of AI

AI was employed as an auxiliary tool in this project, in complete accordance with PRI's published policies, to assist in improving text written by us.