Boardverse: Enhancing Board Game Discovery Through Domain-Specific Search

Lucas Nélson Pinheiro Faria
up202207540@up.pt
FEUP
Porto, Portugal

Pedro Rafael Correia Borges
up202207552@up.pt
FEUP
Porto, Portugal

Alexandre Fernandes Lopes
up202207015@up.pt
FEUP
Porto, Portugal

Abstract

When players seek new board games to explore, the discovery process often depends on fragmented online sources or subjective recommendations, which complicates the identification of games that align with individual preferences. This study addresses this challenge by enabling independent access to structured and detailed information about board games. Data was collected and processed from BoardGameGeek (BGG), consolidating metadata such as game mechanics, categories, designers, publishers, and user ratings. The resulting dataset was examined to understand its composition, providing a foundation for the development of a comprehensive search infrastructure. After examining and preparing the dataset, a set of information needs was defined, and a search system was implemented using Solr. The system was then evaluated against these information needs to assess its ability to address them effectively. In the final stage of the project, a semantic search approach was explored, and several enhancements were introduced, including a hybrid model that integrates both semantic and lexical retrieval, to produce the final version of the board game search system. Additionally, a simple user interface was developed to provide a more intuitive and accessible interaction experience.

Keywords

Information Retrieval, Search Engine, Board Games, Dataset Preparation, Data Exploration, Conceptual Domain, Data, Pipeline, Indexing, Query, Information Need

1 Introduction

Board games have experienced a significant resurgence in popularity over the last two decades. With thousands of titles released every year, ranging from simple family games to complex strategic games, discovering titles that align with individual preferences has become a non-trivial task. Players often rely on fragmented online sources, subjective reviews, or community recommendations, which do not always provide structured or comprehensive information.

To address this challenge, this project focuses on the development of a search and retrieval system for board games, built upon structured data extracted from the BoardGameGeek (BGG) [10] platform. The goal is to enable users to search, filter, and explore games efficiently based on various attributes such as mechanics, categories, designers, publishers, and user ratings.

2 Data Sources

The initial stage of the project focused on identifying a suitable data source for the development of a search system. The selected dataset was required to contain meaningful information, sufficient detail to distinguish individual entries, and a representative volume of records. Given the thematic focus on board games, an online search was conducted, and the BoardGameGeek (BGG) [10] dataset was identified as the most appropriate source. The dataset comprises information sourced directly from game manufacturers, including attributes such as title, description, publisher, and minimum recommended age for play. In addition, it incorporates community-contributed data from the BoardGameGeek [10] platform, such as user ratings, ownership counts, wishlist frequency, and perceived difficulty levels. BoardGameGeek [10] operates as a dynamic, community-driven platform, continuously updated in real time. Registered users can submit new game entries, which are subsequently reviewed and approved by a dedicated team of administrators to ensure data integrity and consistency.

3 Data Pipeline

This section describes the procedures for acquiring, cleaning, and processing the data for our board games search system [Figure 1].

3.1 Data Collection

BoardGameGeek (BGG) [10] offers a comprehensive repository of board game metadata, encompassing attributes such as mechanics, categories, designers, publishers, and user ratings. To construct the dataset, data was collected using a combination of official CSV data dumps and BGG's XML API. This dual-source approach ensured the inclusion of both structured and unstructured textual data, aligning with the requirements of the project.

The CSV data dump `bg_ranks.csv`, provided by the BGG platform, was employed to retrieve basic game identifiers and ranking information. Detailed metadata, including game titles, descriptions, playing times, player counts, minimum age recommendations, and other relevant attributes, was subsequently obtained via the XML API. Requests were issued in batches of 20 games to comply with BGG's rate limitations.

To automate the data collection process, a Python [13] script was developed. The `requests` module facilitated communication with the API, enabling the transmission of queries and reception of XML-formatted responses. These responses were parsed using Python's `xml` module, and the resulting structured data was stored in CSV format via the `csv` module.

Due to file size constraints, particularly those imposed by upload limits in version control software, the resulting CSV file was segmented into smaller files using an additional Python [13] script. This ensured efficient storage and version control while maintaining data accessibility.

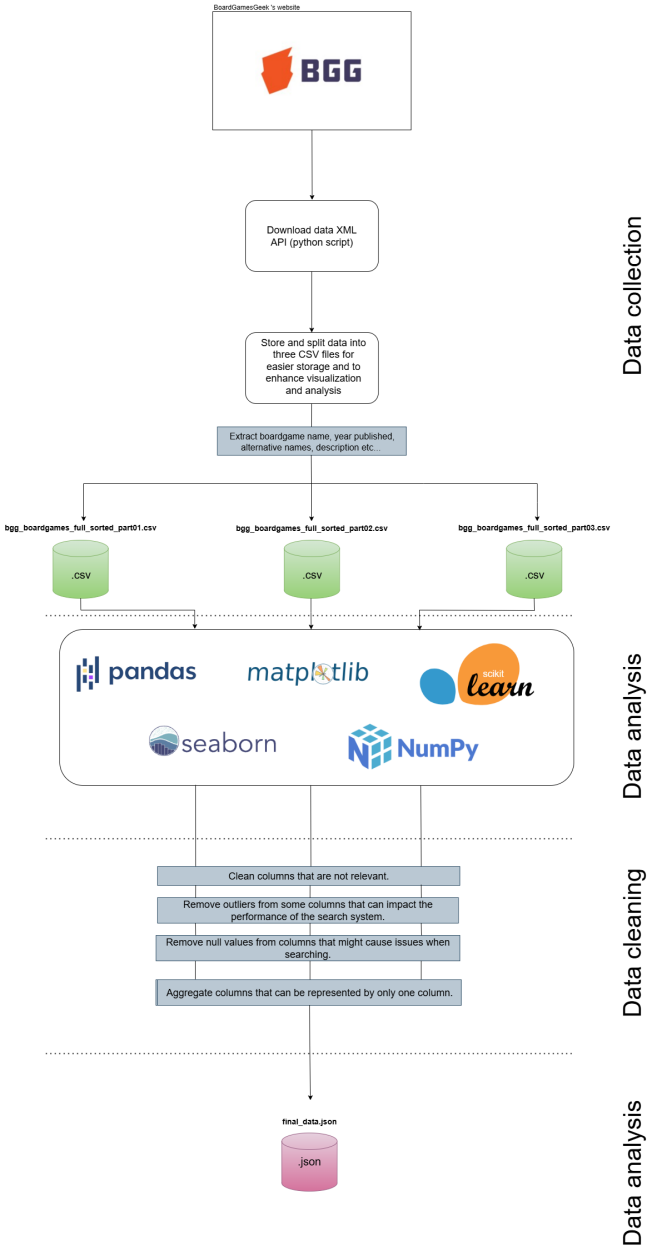

Figure 1: Data Collection and Preparation Diagram.

After this step, the merged dataset, resulting from the combination of the three source files, comprised 132382 rows and 37 columns.

3.2 Data Processing

The raw data required processing to ensure quality and consistency. For this stage, several Python [13] libraries were employed, namely Pandas[6], NumPy [5], Matplotlib [3], and Seaborn[24] to facilitate data manipulation, exploratory analysis, and visualization.

The initial step involved **concatenating** the multiple CSV files generated during the data collection phase into a single Pandas[6] DataFrame. This consolidation simplified subsequent analysis and cleaning procedures by enabling uniform access to all records.

Next, an exploratory **analysis** was conducted to identify patterns, outliers, null values, and irrelevant attributes. Based on these insights, a Python [13] script was developed to perform targeted cleaning operations. The columns "type", "rank_boardgame", "ranks_other", "median", "image", and "thumbnail" were removed. The "type" column [Figure 4] was excluded due to its lack of variability, it contained only the value "boardgame". The "rank_boardgame" column was limited to approximately 33,000 entries, with an unclear ranking methodology. Meanwhile, "ranks_other" and "median" consisted entirely of null values. The "image" and "thumbnail" columns contain only URLs pointing to visual assets associated with each game. As the search system is designed to operate primarily on textual data, these image references were deemed irrelevant and subsequently excluded from the dataset.

Following the initial cleaning phase, extreme **outliers** in several numeric fields were removed to enhance data reliability and consistency. Entries with values exceeding 100 in the "maxplayers" column were excluded, as there are no credible records of board games designed for more than 100 participants [21][18]. Similarly, games with "yearpublished" values greater than 2025 were discarded, given that such entries refer to future dates and likely represent erroneous or placeholder data. Additional filtering was applied to the "playingtime", "maxplaytime", and "minplaytime" columns, where values above 500000 minutes (approximately one year) were considered implausible. While some games may involve extended durations (many times more than 1000 hours [9]), instances exceeding this threshold are exceedingly rare and likely reflect data entry errors. To ensure logical consistency, games where "minplaytime" exceeded "maxplaytime" were also removed. Finally, entries with "minage" values above 21 were excluded, as this surpasses the legal adulthood threshold in most countries and is unlikely to represent valid age recommendations for board game participation [1].

The next step involved removing rows containing **null values**, focusing specifically on columns deemed essential for identifying and characterizing board games. These key attributes included "description", "maxplayers", "minplayers", "yearpublished", "publishers", "categories", and "minage", as they represent core metadata provided by manufacturers. Among these columns, "minage" [Figure 6] and "yearpublished" [Figure 7] exhibited the highest proportion of missing values, whereas "description" and "publishers" had less than 1% null entries. This selective filtering ensured the retention of records with sufficient descriptive and structural information for downstream analysis.

The final stage of data processing involved **aggregating** the "minplaytime" and "maxplaytime" columns into a single "playingtime" attribute, representing the average duration of a game session. In the original dataset, some entries included only "minplaytime" and "maxplaytime", while others provided values exclusively for "playingtime". To enhance consistency across records, a conditional imputation strategy was applied: for rows where "playingtime" was null, the mean of "minplaytime" and "maxplaytime" was calculated and assigned to the "playingtime" column. This approach ensured that all entries retained a standardized measure of play

duration. Following this aggregation, the original "minplaytime" and "maxplaytime" columns were removed, leaving "playingtime" as the unified and representative metric.

Upon completion of the data cleaning process, the refined dataset was exported in JSON format, ensuring structured access to the information.

The entire processing workflow is thoroughly documented and fully reproducible, enabling seamless future updates, extensions, or integration into other systems.

4 Data Characterization

After the processing phase, the final dataset comprised **96025** board game entries and **31** columns. To characterize the data, a combination of analytical and visualization tools was employed. Matplotlib [3] and Seaborn [24] were used to generate plots and visual summaries, while Scikit-learn [7] facilitated keyword extraction through TF-IDF scoring. Additionally, Pandas [6] and NumPy [5] supported efficient data manipulation and statistical analysis.

Key characteristics identified include:

- **Textual richness**: Each entry includes a detailed description, categories, mechanics, and designer information, suitable for textual search and retrieval. Some of the most important words found in the description are related to the board game thematic like "game", "card" and "board" [Table 1].
- **Structured metadata**: Numerical attributes such as `minplayers`, `maxplayers`, `playingtime`, `minage`, and user ratings allow filtering and faceted search.
- **Data distribution**: Most games have playing times under 100 minutes, a minimum age between 3 and 18 years, a maximum of 6 players and were published in the 21st century [Figure 5][Figure 2].
- **Publisher diversity**: Over 33925 unique publishers are represented, with a long tail of publishers having only a few games.
- **Correlation between columns**: The average user rating exhibits a positive correlation with several engagement-related metrics, including the number of owners, the number of wishlists a game appears in, and the number of users interested in trading the game. Pearson correlation coefficients [22] for these relationships range from 0.64 to 0.84, indicating moderate to strong linear associations. The correlation is 0.64 with both the number of owners [Figure 8] and the number of trades [Figure 11], 0.77 with wishlist frequency [Figure 9], and 0.84 with the count of users expressing interest in trading [Figure 10]. These results suggest that higher-rated games tend to attract greater user interaction and visibility within the community.

Table 1: Ranking of key words by TF-IDF (excluding plurals).

Rank	Word	Score
1	game	0.588
2	player	0.348
3	card	0.233
4	play	0.145
5	board	0.113

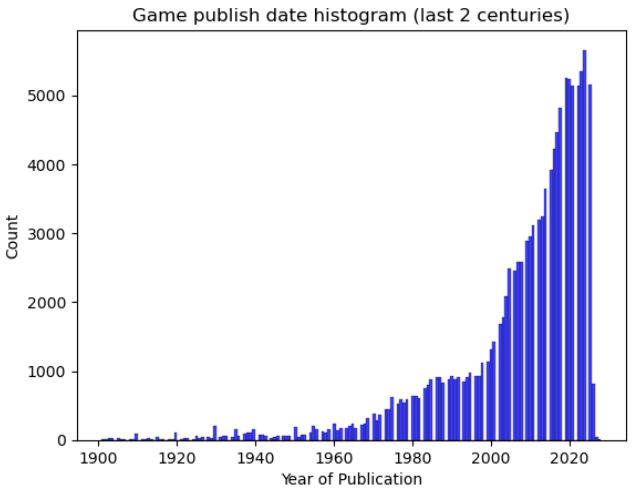

Figure 2: Distribution of games per release year (last two centuries).

5 Conceptual Domain and Document Definition

Each board game entry is treated as a *document* in the information retrieval system. The document structure includes:

- **Title**: The primary name of the game.
- **Description**: Free-text field summarizing the game, its mechanics, and themes.
- **Metadata fields**: Categories, mechanics, designers, publishers, number of players, playing time, and minimum age.
- **Rating and popularity**: User ratings and BGG rank information.

This conceptual model [Figure 3] enables multi-faceted queries and supports relevance ranking based on textual and numerical attributes.

6 Prospective Search Tasks

Based on the dataset, prospective search tasks include:

- **Keyword search**: Allowing users to find games by name, description, or mechanics.
- **Faceted filtering**: Users can filter games by the number of players, playing time, age, categories, or publisher.
- **Recommendation-style ranking**: Ranking results based on user ratings, popularity, or similarity to other games.

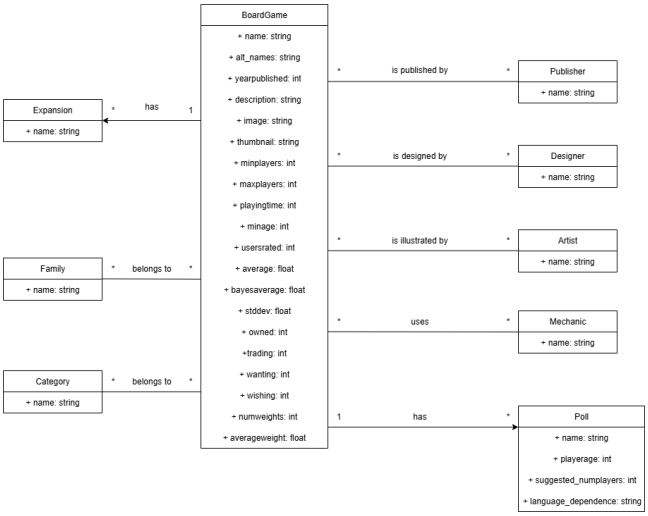

Figure 3: Conceptual Model Diagram.

These search tasks guide the design and evaluation of the retrieval system in subsequent milestones.

Some possible examples of search scenarios that make use of the search tasks include:

- "A group of 4 friends loves complex strategy games with engine-building mechanics, but they dislike deck-building. They want a game that lasts 90–120 minutes, is suitable for exactly 4 players, and has a fantasy or sci-fi theme. Additionally, they prefer games that are highly rated by reviewers, but not from the same publisher they already own."
- "Someone is hosting a party with 6–8 attendees ranging from teenagers to adults. They want fast, social, humorous games that are easy to learn, allow team play, and last 20–40 minutes each. Games should be non-competitive enough that beginners don't feel frustrated but still engaging for experienced players."
- "A hobbyist wants a game that is either a deck-builder or tableau-builder, cooperative, plays 2–4 players, and has a thematic story component. They want the average playtime to be 60–90 minutes, and the game should allow casual and serious players to enjoy it together."

7 Information Retrieval

Information Retrieval (IR)[23] is a field of computer science concerned with retrieving relevant information in response to user queries. It can be defined as the process of locating unstructured data within a collection to satisfy a specific information need. Retrieved results are presented to users in order of relevance, determined by computed ranking scores.

This section describes the indexing and retrieval processes implemented in the proposed system, which builds upon the previously collected and processed dataset. The system is powered by Apache Solr[16], an open-source, multi-modal search platform that enables efficient indexing and retrieval. Solr[16] is built on Apache Lucene[15], providing advanced capabilities for full-text, vector-based, and geospatial search.

7.1 Document Definition

The documents used in the system are derived from the data collection and processing steps outlined in the previous sections. Each board game, as described in Section 5, is represented as a document within the information retrieval system. These documents include the game's title, description, metadata fields, and popularity metrics. This structured organization enables Apache Solr[16] to efficiently index and retrieve information, ensuring that queries can be answered with relevant and well-ranked results.

7.2 Indexing Process and Schema Definition

The indexing process in information retrieval (IR) systems leads to the construction of an IR index, a data structure derived from a document collection that facilitates efficient and effective query processing. Indexing typically involves parsing documents to extract representative features and organizing them into searchable structures. A possible approach is the inverted index, in which each term is associated with the identifiers of the documents in which it occurs. Another possible used representation is the vector space model, such as the bag-of-words [23].

In Solr, a schema can be defined to guide both the indexing and query processes. The schema specifies the structure of documents by creating fields and configuring their properties. Each field can be customized with an appropriate type and attributes, ensuring accurate data representation and enabling more efficient indexing. A well-designed schema enhances both the performance of the system and the relevance of search results [12].

In the specific case of the board games search system, two distinct schemas were used: one more simple using only the default Solr values and a more complex schema with custom fields. In the board games search system, 13 out of the 31 fields are textual. Each of these fields exhibits distinct characteristics that must be carefully addressed to ensure optimal performance during both indexing and querying. To achieve this, three new field types were defined, tailored to the dataset's requirements and the retrieval tasks of the system:

- **text_boost**: This field type is applied solely to the description field and aims to enhance recall by generating broader search results, reflecting the nature of the field's extended text segments. This field is defined as a Solr TextField with a custom analyzer. The analyzer employs Solr's standard tokenizer to segment text into tokens and applies a sequence of five filters to normalize and enrich the data. First, the lowercase filter converts all characters to their lowercase form, ensuring case-insensitive matching. Next, the Porter stemming filter reduces words to their root form, improving recall by grouping morphological variants under a common stem. A synonym filter is then applied to expand tokens with predefined equivalences, enhancing query flexibility. The stop word filter removes frequent but semantically uninformative terms, thereby reducing noise in the index. Finally, the ASCIIFolding filter normalizes accented characters to their ASCII

equivalents, ensuring consistent handling of multilingual or diacritic-rich text.

- **text_exact_boost**: This field type is applied to fields that require more exact matching, including game names, publishers, designers, and artists. It is implemented as a Solr TextField using the standard tokenizer and a sequence of three filters. The first filter converts all tokens to lowercase, ensuring case-insensitive matching. The second applies minimal English stemming, reducing words to their base form while preserving precision. Finally, the ASCIIFolding filter normalizes accented characters to their ASCII equivalents.
- **text_exact_extra_boost**: This field type builds upon the previous configuration by incorporating two additional filters to broaden the scope of search results. It is intended for fields that require a certain level of exactness, such as mechanics, categories or families, but can also benefit from matching related or similar expressions. Compared to the earlier type, this configuration introduces a synonym filter, which expands tokens based on predefined equivalences, and a stop word filter, which removes frequent but semantically uninformative terms. Together, these enhancements strike a balance between precision and flexibility, enabling more comprehensive retrieval in contexts where exact matches alone may be insufficient.

To generate a comprehensive list of synonyms and stop words for use with the filters applied in the newly defined field types, two Python[13] scripts were developed. The first script extracted nouns from the textual fields and identified their synonyms using the NLTK[4] library in combination with the WordNet extension. The second script focused on detecting stop words present in the textual fields, employing NLTK[4] together with the Punkt tokenizer and the Stopwords corpus. These scripts ensured that the synonym and stop word lists were tailored to the dataset, thereby improving the effectiveness of the text analysis pipeline. The synonym and stop word lists were subsequently stored in separate files and deployed into the container hosting the Solr[16] instance.

As previously mentioned, the simplest schema relies solely on Solr's default field types, whereas the more advanced schema introduces custom types. Fields related to player polls are not indexed, since they do not contribute to satisfying any information need, as will be demonstrated in Section 8. Conversely, fields containing evaluation metrics—such as the standard deviation of scores and the number of voters—are not stored, because they are not relevant to any retrieval task. Finally, all fields that can contain multiple values are defined as multivalued, including publishers, designers, artists, and similar attributes. Some numeric fields also have a string field variation to allow direct search on those fields. A more complete description can be seen in the tables 2 and 3.

Table 2: Simple Schema Fields.

Field	Type	Indexed
name	text_general	true
alt_names	text_general	true
description	text_general	true
yearpublished	plongs	true
yearpublished_str	string	true
minplayers	plongs	true
minplayers_str	string	true
maxplayers	plongs	true
playingtime	plongs	true
playingtime_str	string	true
minage	plongs	true
minage_str	string	true
publishers	text_general	true
designers	text_general	true
artists	text_general	true
categories	text_general	true
mechanics	text_general	true
families	text_general	true
expansions	text_general	true
poll_suggested_numbplayers	text_general	false
poll_playerage	text_general	false
poll_language_dependence	text_general	false
usersrated	plongs	true
average	plongs	true
bayesaverage	plongs	true
stddev	plongs	true
owned	plongs	true
trading	plongs	true
wanting	plongs	true
wishing	plongs	true
numweights	plongs	true
averageweight	plongs	true

7.3 Retrieval Process

As mentioned in the previous section, two schemas were used: one simple using only default Solr[16] types and a more complex schema with costume types. To query the data, in both schemas, several techniques were used to retrieve the best results. The queries were made using the JSON API provided by Solr. The main parameters used were:

- **Query (q)**: Where the main words of the query are placed.
- **Fields**: The fields returned in the response. This parameter was only used to retrieve the most important fields for a given information needs, enabling a quicker testing.
- **Sort**: The sort parameter orders the retrieved documents based on the value of a specified field. In this configuration, sorting was applied only to the score field, ensuring that the query structure remains generalized and adaptable to any information need.
- **Limit**: The limit parameter restricts the number of retrieved documents by returning only the top x results. To streamline the evaluation process, a fixed limit of 30 documents was applied to each query.
- **Query Operation (q.op)**: The "AND" operation was applied to combine all terms in the q parameter, ensuring that the retrieved results contain every input specified by the user.

The deftype used in the more advanced queries, applied only in the complex schema, was eDismax, allowing the use of more parameters to enhance the search results:

- **Query Fields with Boosts (qf)**: Boosting allows certain fields to be given higher relevance than others during query evaluation. In this configuration, the fields and their boost

Table 3: Complex Schema Fields.

Field	Type	Indexed
name	text_exact_boost	true
alt_names	text_exact_boost	true
description	text_boost	true
yearpublished	pint	true
yearpublished_str	string	true
minplayers	pint	true
minplayers_str	string	true
maxplayers	pint	true
playingtime	pint	true
playingtime_str	string	true
minage	pint	true
minage_str	string	true
publishers	text_exact_boost	true
designers	text_exact_boost	true
artists	text_exact_boost	true
categories	text_exact_extra_boost	true
mechanics	text_exact_extra_boost	true
families	text_exact_extra_boost	true
expansions	text_exact_extra_boost	true
poll_suggested_numbplayers	string	false
poll_playerage	string	false
poll_language_dependence	string	false
usersrated	plong	true
average	pfloat	true
bayesaverage	pfloat	true
stddev	pfloat	true
owned	plong	true
trading	plong	true
wanting	plong	true
wishing	plong	true
numweights	plong	true
averageweight	pfloat	true

values were defined as follows: *name^5 alt_names^2 description^4 categories^4 mechanics^4 publishers^4 designers^3 families^2 expansions^2 minage_str^3 yearpublished_str^3 playingtime_str^3 minplayers_str^3*. The *name* field carries the highest weight, ensuring that queries targeting game titles are matched with maximum precision. The *categories*, *mechanics*, and *publishers* fields are the next most influential, as they capture the core characteristics of each game and support thematic or feature-based searches. Fields such as *designers* and the numeric attributes (e.g., minimum age, year published, playing time, and minimum players) have medium importance. While not as dominant, they are valuable for filtering games based on specific requirements, such as the number of players or age suitability. Finally, *families* and *expansions* are assigned lower boost values. These fields provide complementary details that enrich the search results, but they are secondary compared to the main descriptive and categorical fields.

- **Phrase Boost Field (pf)**: Phrase boosting increases the relevance of documents where query terms appear together as a phrase within specific fields. In this configuration, the phrase boost values were defined as: *name^4 description^2 alt_names^2 publishers^4 designers^2 artists^2 categories^4 mechanics^4 families^2*. This setup assigns greater importance to the *name*, *publishers*, *categories*, and *mechanics* fields, as they provide the most critical information about board games and directly address typical information needs. The remaining fields, such as *description*, *alt_names*, *designers*, *artists*, and *families*, contribute additional context and detail, but with lower boost values to reflect their secondary role in supporting the search process.
- **Phrase Slop (ps)**: The phrase slop parameter defines the allowable variation in word order or the distance between terms that Solr[16] accepts when matching phrases. In this configuration, a slop value of 2 was applied, permitting slight flexibility in word positioning while still maintaining precise and relevant results.
- **Boost Function (bf)**: The boost function applies a multiplicative adjustment to the query score, enhancing the ranking of documents based on specific numeric attributes. In this configuration, the following boost functions were used: *recip(bayesaverage,1,10,10)^3 recip(owned,1,1000,1000)^1 recip(trading,1,1000,1000)^1 recip(wanting,1,1000,1000)^1 recip(wishing,1,1000,1000)^3*. The *recip* function applies a reciprocal transformation to field values, ensuring that higher values contribute positively to the score but with diminishing returns. This means that initial increases in a field, such as ownership count or rating, have a strong impact, while extremely large values do not dominate the ranking. In practice, this configuration prioritizes games with higher community ratings through the *bayesaverage* field, giving them greater relevance in the results. At the same time, it promotes games that are popular among players, reflected in ownership counts and the number of users wishing or wanting the game. By combining quality indicators with popularity signals, the boost function balances precision with community interest, ensuring that the most highly evaluated and widely appreciated board games appear more prominently in search results.

Another technique explored was the use of term boosts, which increase the importance of specific terms within a query. However, this approach was not included in the final query structure, as it does not support a generalized configuration that can be applied consistently across all queries. Query filters were also considered, but ultimately excluded for the same reason: they prevent the creation of a flexible, text-based query structure suitable for diverse information needs. Finally, wildcards, fuzziness, and proximity searches on specific terms were tested but not extensively adopted, since they did not lead to meaningful improvements in retrieval performance for the evaluated queries.

An example of a query is present in the table 4.

Table 4: Query structure example for the complex schema.

Parameter	Value
limit	30
fields	id,score, name,description,mechanics, categories, families
defType	edismax
sort	score desc
q	strategy deck building
q.op	AND
qf	name^5 alt_names^2 description^4 categories^4 mechanics^4 publishers^4 designers^3 artists^1 families^2 expansions^2 minage_str^3 yearpublished_str^3 playingtime_str^3 minplayers_str^3
pf	name^4 description^2 alt_names^2 publishers^4 designers^2 artists^2 categories^4 mechanics^4 families^2
bf	recip(bayesaverage,1,10,10)^3 recip(owned,1,1000,1000)^1 recip(trading,1,1000,1000)^1 recip(wanting,1,1000,1000)^1 recip(wishing,1,1000,1000)^3
ps	2

8 Evaluation

Evaluation[11] of retrieval systems is a critical step in information retrieval, as it provides objective proof of how a system meets user information needs. In practice, evaluating retrieval systems is challenging because relevance is inherently subjective; different users may have varying expectations about what constitutes a relevant result. To mitigate these biases, evaluation relies on well-defined information needs, carefully constructed relevance judgments, and standard metrics that approximate user satisfaction. Metrics such as **Average Precision (AvP)**, **Precision at rank *k* (P@k)**, and **Mean Average Precision (MAP)** are widely adopted to compare system effectiveness. AvP accounts both precision and recall across ranked results, P@k focuses on the top k results which are most visible to users, and MAP provides a single measure aggregating AvP across multiple queries.

8.1 Evaluation Metrics

- **Average Precision (AvP)**: Measures the average precision across all relevant documents for a single query, rewarding systems that rank relevant documents higher. From a user perspective, higher AvP implies that relevant results are usually among the top ranks, reducing effort and increasing overall user satisfaction.
- **Precision at *k* (P@k)**: Measures the distribution of relevant items within the first k results. Since users normally only inspect the top results, P@k aligns closely with observed user behavior.
- **Mean Average Precision (MAP)**: MAP aggregates AvP across all queries. It is widely used in IR to quantify overall system consistency. A higher MAP means that typical users, across varied information needs, experience reliable retrieval quality.

These metrics help reduce subjectivity by grounding evaluation on clearly defined relevance judgments. Relevance was manually assigned for each query. Documents were labeled as relevant (1) or not relevant (0), ensuring consistency across the evaluation.

8.2 Per-Information-Need Evaluation

8.2.1 Information Need: Strategy Games with Deck-Building Mechanics.

- **Query (p):** strategy deck building.
- **Relevance Judgment:** Documents were only marked as relevant if they explicitly included both the strategy category and deck-building mechanics.

Table 5: Q1 information need metrics.

System	AvP	P@30
Simple	0.8620	0.6667
Complex	0.9868	0.9333

- **Analysis:** Both systems retrieve relevant documents effectively. The complex system achieves near-perfect results, demonstrating that structured scoring and boosted fields improve ranking consistency. The simple system retrieves relevant items effectively but ranks some less relevant items higher, slightly reducing top-ranked precision. The P–R (precision-recall) curve for the simple schema [Figure 12] shows high initial precision that gradually decreases as recall increases. In contrast, the P–R curve for the complex schema [Figure 13] maintains high precision even at higher recall levels, demonstrating more stable performance.

8.2.2 Information Need: Game for 2 players that has a fantasy theme with dice rolling mechanic.

- **Query (p):** Fantasy dice rolling 2 players.
- **Relevance Judgment:** Documents were marked relevant only if they explicitly included two-players family, fantasy category, and dice-rolling mechanic.

Table 6: Q2 information need metrics.

System	AvP	P@30
Simple	0.6793	0.4000
Complex	0.5291	0.5000

- **Analysis:** The simple system retrieved only 19 results, which inflated average precision since most items were relevant. The complex system, by retrieving more documents, improved P@30 but introduced less relevant results, lowering AvP. This highlights how query specificity and document coverage influence ranking performance. The P–R curves further illustrate this difference: the simple system [Figure 14] begins with high precision that quickly declines as recall increases, while the complex system [Figure 15] starts lower but maintains more stable precision across higher recall levels.

8.2.3 *Information Need: Card Games published by Hasbro.*

- **Query (p):** Card Games Hasbro.
- **Relevance Judgment:** Documents were only marked as relevant if they explicitly included both the card game category and the publisher Hasbro.

Table 7: Q3 information need metrics.

System	AvP	P@30
Simple	0.4642	0.4333
Complex	0.5265	0.7000

- **Analysis:** The complex system achieves higher P@30, ranking more Hasbro card games among the top results. In contrast, the simple system maintains comparable AvP, reflecting fewer irrelevant documents overall. This variation highlights how broader scoring in the complex system can elevate top-ranked results while introducing some noise. The P–R curves reinforce this difference: the simple system [Figure 16] begins with high precision that quickly declines as recall increases, whereas the complex system [Figure 17] starts lower but sustains precision more consistently, demonstrating greater stability across recall levels.

8.2.4 *Information Need: Card games released in 2020 or 2021 with cooperative or deck-building mechanics.*

- **Query (p):** (card AND (cooperative OR deck building) AND (2020 OR 2021)).
- **Relevance Judgment:** Documents were marked relevant if they included the card game category, were released in 2020 or 2021, and had deck-building mechanics.

Table 8: Q4 information need metrics.

System	AvP	P@30
Simple	0.4580	0.3667
Complex	0.3948	0.3000

- **Analysis:** In this restrictive information need, the simple system outperforms the complex one in AvP. The complex system introduces noise due to broader token-proximity handling and the ranking of partially matching documents, showing that complex scoring does not always yield better performance for highly specific queries. The P–R curves confirm that both systems performed worse compared to other information needs. Moreover, the complex system [Figure 19] exhibits a steeper decline in precision than the simple system [Figure 18], underscoring its lower performance in this scenario.

8.3 Global results

Taking all results into account, the Mean Average Precision for both systems is:

Table 9: Global Mean Average Precision.

Global	MAP
Simple	0.6159
Complex	0.6093

Both systems perform satisfactorily, with slightly different strengths. The complex system excels in ranking relevant documents higher for structured queries, improving user efficiency in browsing. The simple system provides more consistent relevance across restrictive queries, minimizing irrelevant results. Overall, from a user perspective, the choice between systems depends on whether top-ranked results or overall reliability is more important.

9 Information Retrieval Improvements

During the previous stage, two systems were developed. The first system relied on a simple schema using Solr's default field types and a basic query structure, without incorporating advanced features such as boosts. The second, more advanced system introduced custom field types and a richer query structure that included boosted query fields, phrase boosts, phrase slop parameters, and boost functions. It also employed a more sophisticated tokenizer configuration, incorporating filters such as Porter stemming, lowercase normalization, synonym expansion, and stop-word removal.

The previous stage revealed several opportunities for improving the quality of the results. The complex system did not yield significant gains in overall performance, indicating the need for further refinement. In this next stage, the advanced system serves as the foundation for additional enhancements aimed at improving retrieval effectiveness. Two main approaches were explored: a semantic search method, designed to produce results that better reflect the meaning of the query and the game descriptions, and a hybrid model that combines the strengths of both semantic and lexical search into a unified retrieval system.

9.1 Semantic Search

Semantic search[17] is a data retrieval technique that focuses on understanding the meaning and context behind a user's query, rather than relying solely on exact keyword matches. It evaluates how well document content aligns with the query's intent.

In Solr, semantic search can be implemented using dense vector fields. Unlike traditional term-based sparse vector representations, dense vectors encode approximate semantic meaning into a fixed number of dimensions. This dimensionality is typically much lower than in sparse representations while still capturing richer semantic relationships [14].

To enable dense-vector-based retrieval in the system, a new field with a custom type was added to the **schema definition**. The field, named combined_vector, is both indexed and stored. It uses a custom type called gameVector, based on Solr's DenseVectorField class, with a vector dimension of 384, cosine as the similarity function,

and HNSW as the kNN search algorithm. All other fields remain consistent with those used in the complex schema described in the previous stage, shown in Table 3.

Unlike the traditional sparse-vector approach, dense vector generation cannot be performed directly within Solr and must instead be handled externally in the application logic. Producing dense vector representations requires a deep learning model outside of Solr capable of encoding textual information into fixed-dimension embeddings [14]. For this purpose, a Python script was developed using the Sentence Transformers library [8] and the all-MiniLM-L6-v2 model [2]. This model maps sentences and paragraphs into 384-dimensional dense vectors, the same dimensionality defined for the corresponding field in the Solr schema. All relevant textual fields in the dataset were used to generate embeddings, including description, name, alt_names, publishers, designers, categories, mechanics, and families. Given the size of the dataset (close to one hundred thousand documents), GPU acceleration was required to make the embedding process feasible. By enabling the "cuda" option, the model was executed on an NVIDIA graphics card using NVIDIA's CUDA platform[20], reducing the embedding generation time from several hours to just a few minutes.

For the **retrieval process**, each query must first be converted into an embedding using the model described in the previous paragraph. The query structure for the semantic search system is simpler than in the complex system because it does not require field boosts or phrase boosts. Since semantic search aims to retrieve text that is conceptually related to the query rather than matching specific terms, such boosts provide no meaningful benefit in this context. The "topk" parameter was set to 100, meaning that the system retrieves the 100 documents most semantically related to the query. An example of the structure of the query for the semantic search can be found in the table 10.

Table 10: Query structure example for the semantic schema.

Parameter	Value
fields	id,score, name,description,mechanics, categories, families
q	`{!knn f=combined_vector topK=100}(embedding)`

9.2 Hybrid System

Semantic search and lexical search each present distinct advantages and limitations. Semantic search excels at retrieving documents that are conceptually related to the query, even when they do not share explicit vocabulary. However, it may also return irrelevant results when the query lacks sufficient context or when the text used to generate embeddings is ambiguous. Conversely, lexical search is effective at retrieving documents containing exact word matches or recognized synonyms, but it lacks an understanding of semantic meaning. As a result, it may surface documents that contain the correct keywords yet fall outside the intended context.

To combine the strong points of both semantic search and lexical search, a hybrid search was built. It used the complex schema as base and adds the new field "combined_vector" created for the semantic schema. The query structure used both components from the semantic search and from the lexical search.

However, Solr provides no official documentation on hybrid search, aside from a single blog post referenced in the Solr documentation. This lack of formal guidance made the construction of a hybrid query considerably more challenging.

The chosen approach relies on two main components. The first component reuses the fields employed in the complex system's queries (as shown in Table 4), with the addition of the "knn" field to incorporate the semantic search parameters. These parameters include the target field in the schema, the query embedding vector, and the value of k, which determines how many semantically related documents should be retrieved. This component effectively returns the union of documents that would be obtained from both the lexical and semantic search processes. The second component focuses on ranking the retrieved results. Because lexical scoring typically produces much larger score values, the top results would otherwise be dominated by the lexical component, replicating the behavior of a purely lexical search. To mitigate this, a custom scoring function was implemented. It normalizes the lexical scores to the same range as the semantic scores, applies a weighting of 70% to the semantic score and 30% to the lexical score, and then combines them. The final ranking therefore reflects a balanced contribution from both lexical and semantic relevance. An example of the query structure is shown in Table 11.

Table 11: Query structure example for the hybrid schema.

Parameter	Value
fields	id, score, name, description, mechanics, categories, families
defType	edismax
q	strategy deck building
q.op	AND
qf	name^5 alt_names^2 description^4 categories^4 mechanics^4 publishers^4 designers^3 artists^1 families^2 expansions^2 minage_str^3 yearpublished_str^3 playingtime_str^3 minplayers_str^3
pf	name^4 description^2 alt_names^2 publishers^4 designers^2 artists^2 categories^4 mechanics^4 families^2
bf	recip(bayesaverage,1,10,10)^3 recip(owned,1,1000,1000)^1 recip(trading,1,1000,1000)^1 recip(wanting,1,1000,1000)^1 recip(wishing,1,1000,1000)^3
ps	2
knn	`field = combined_vector vector = embeddings k = 100`
lexicalQuery	`{edismax q.op=AND qf='name^5 alt_names^2 description^4 categories^4 mechanics^4 publishers^4 designers^3 artists^1 families^2 expansions^2'}(strategy deck building)`
vectorQuery	`{knn f=combined_vector topK=100}(embedding)`
sort	`{func} sum(product(0.3,scale(query($lexicalQuery),0,1)), product(0.7,query($vectorQuery))) desc`

10 Evaluation on the Improved Systems

To assess how the improvements affect the system's ability to meet users' information needs, an additional evaluation step was conducted. As in the previous stage of the project, the results were evaluated using the metrics **Average Precision (AvP)**, **Precision at rank k (P@k)**, and **Mean Average Precision (MAP)**. All systems were tested using the same query text to ensure a fair comparison. However, because semantic search benefits from additional context, each query was also evaluated using more expressive, natural-language formulations rather than relying solely on isolated keywords. The lexical complex system developed in the previous stage is also included in the evaluation so it can be used as a baseline for comparison.

It is important to note, as discussed in Section 8, that the evaluation of retrieval systems is inherently subjective. Different users may have different expectations about what constitutes a relevant result, meaning that the outcomes presented here could vary if the evaluation were carried out by another group of people.

10.1 Per-Information-Need Evaluation on the Improved Systems

10.1.1 Information Need: Strategy Games with Deck-Building Mechanics.

- **Query(q):** strategy deck building.
- **Expressive Query(q):** Strategy games with deck mechanics.
- **Relevance Judgment:** Documents were only marked as relevant if they explicitly included both the strategy category and deck-building mechanics.

Table 12: Q1 information need metrics improved systems.

System	AvP	P@30
Lexical (complex)	0.9868	0.9333
Semantic Keywords	0.2681	0.2667
Semantic Expressive	0.5629	0.3333
Hybrid Keywords	0.8355	0.8333
Hybrid Expressive	0.247	0.3333

- **Analysis:** When compared to the complex system from the previous stage, all newly tested systems performed worse. The semantic search model showed weak performance when using only keywords, but its results improved significantly with more expressive queries. This outcome is expected, since semantic search relies on the contextual richness of the query to identify documents with similar meaning. The hybrid system, on the other hand, performed relatively well with keyword-only queries, showing only a slight decrease in both metrics. However, its performance dropped considerably when using expressive queries. Because the hybrid approach combines both semantic and lexical components, these results suggest that this particular information need is better aligned with a primarily lexical strategy. This is further supported by the stronger performance of the keyword-based variant, which naturally emphasizes the lexical component. The P–R curves reinforce these observations. Semantic search with keywords [Figure 20] begins at a moderate precision but drops rapidly as recall increases, whereas semantic search with expressive queries [Figure 24] starts with high precision yet also declines sharply. In contrast, the hybrid system using keywords [Figure 28] maintains strong precision even at higher recall levels, showing only minor decreases. The hybrid system with expressive queries [Figure 32], however, exhibits a more stable but consistently lower precision across the curve.

10.1.2 Information Need: Game for 2 players that has a fantasy theme with dice rolling mechanic.

- **Query(q):** Fantasy dice rolling 2 players.
- **Expressive Query(q):** Game for 2 players that has a fantasy theme with dice rolling mechanic.
- **Relevance Judgment:** Documents were marked relevant only if they explicitly included two-players family, fantasy category, and dice-rolling mechanic.

Table 13: Q2 information need metrics improved systems.

System	AvP	P@30
Lexical (complex)	0.5291	0.5
Semantic Keywords	0.1625	0.0667
Semantic Expressive	0.2808	0.2667
Hybrid Keywords	0.5854	0.5
Hybrid Expressive	0.4845	0.2333

- **Analysis:** For this information need, the hybrid system using keywords was the best-performing approach. It achieved a slightly higher average precision than the purely lexical system. The hybrid system with expressive queries showed a similar average precision but a much lower P@30. The semantic system performed the worst in both variants, although it obtained better results when using expressive queries. These findings indicate that this query is better suited for lexical search. However, combining lexical and semantic signals in the hybrid system still produced slightly better results than using lexical search alone. This behavior is also reflected in the P–R curves. Semantic search with keywords [Figure 21] starts with low precision and remains relatively flat, while the expressive-query variant [Figure 25] also maintains a stable curve but at a higher precision level. The hybrid system with keywords [Figure 29] begins with very high precision, then experiences a sharp drop as recall increases, after which it follows a curve similar to the lexical system. The hybrid system with expressive queries [Figure 33] also starts with very high precision but undergoes an even steeper decline, eventually resembling the curve of semantic search with expressive queries. Overall, these patterns suggest that the keyword-based query places more weight on the lexical component, whereas the expressive-query versions rely more heavily on the semantic component.

10.1.3 *Information Need: Card Games published by Hasbro.*

- **Query(q):** Card Games Hasbro.
- **Expressive Query(q):** Card Games published by Hasbro.
- **Relevance Judgment:** Documents were only marked as relevant if they explicitly included both the card game category and the publisher Hasbro.

Table 14: Q3 information need metrics improved systems.

System	AvP	P@30
Lexical (complex)	0.5265	0.7
Semantic Keywords	0.4514	0.1
Semantic Expressive	0.5526	0.0667
Hybrid Keywords	0.8456	0.9333
Hybrid Expressive	0.9933	0.8333

- **Analysis:** For this information need, the hybrid system proved to be the best performer in both of its variants. While the keyword-based version achieved a higher P@30, the expressive-query variant obtained a better average precision. The lexical and semantic systems showed similar average precision values, but the semantic system had a much lower P@30, indicating greater difficulty in ranking relevant documents in the top 30. Overall, the results suggest that this information need benefits from combining semantic and lexical signals, since using either approach in isolation yields only moderate performance. The P–R curves further support this conclusion: both semantic variants [Figure 22][Figure 26] start with very high precision but experience a sharp decline as recall increases, with the drop occurring even faster for the keyword-based version. In contrast, the hybrid system in both variants [Figure 30][Figure 34] maintains very high precision even at higher recall levels.

10.1.4 *Information Need: Card games released in 2020 or 2021 with cooperative or deck-building mechanics.*

- **Query(q):**(card AND (cooperative OR deck building) AND (2020 OR 2021)).
- **Expressive Query(q):** Card games released in 2020 or 2021 with cooperative or deck-building mechanics.
- **Relevance Judgment:** Documents were marked relevant if they included the card game category, were released in 2020 or 2021, and had deck-building mechanics.

Table 15: Q4 information need metrics improved systems.

System	AvP	P@30
Lexical (complex)	0.3948	0.3
Semantic Keywords	0.1917	0.1333
Semantic Expressive	0.0714	0.0333
Hybrid Keywords	0.2956	0.4
Hybrid Expressive	0.1173	0.1333

- **Analysis:** This information need produced poor results across all systems. The lexical system achieved the highest average precision, while the hybrid system using keyword queries obtained the best P@30. The semantic system performed the worst overall, and unlike in the other information needs, the expressive-query variant performed even worse than the keyword-based one. These results can be explained by the nature of the query. It requires identifying games released in specific years, and because embeddings were generated only from textual fields, the semantic system struggles to interpret numerical constraints that are not well represented in those fields. As a result, it has difficulty establishing meaningful relationships between the query context and the documents. Even for the lexical system, however, this query is challenging due to the multiple constraints it imposes. The P–R curves reflect this behavior. The semantic system with keywords [Figure 23] begins with moderate-to-low precision and quickly drops to a low value, while the expressive-query [Figure 27] variant remains consistently low throughout. In contrast, the hybrid system shows relatively stable curves in both variants: moderate precision for the keyword version [Figure 31] and very low precision for the expressive version [Figure 35].

10.2 Global Results on the Improved Systems

Taking all the results into account, the Mean Average Precision for the improved systems is:

Table 16: Global Mean Average Precision on the Improved Systems.

Global	MAP
Lexical (complex)	0.6093
Semantic Keywords	0.2364
Semantic Expressive	0.3669
Hybrid Keywords	0.6405
Hybrid Expressive	0.4605

The hybrid system using keyword queries proved to be the best overall performer. Although it was slightly worse on the first information need when compared to the lexical system, its results were still very strong. In the fourth query, it achieved lower average precision but obtained the highest P@30, demonstrating its ability to retrieve more relevant documents than the lexical system. The expressive-query variant performed between the lexical and semantic systems, yielding moderate results. Overall, these findings highlight the versatility of the hybrid approach. It is capable of handling both keyword-based and expressive queries with solid performance across different information needs.

11 User Interface

A simple user interface was implemented to improve the overall experience of interacting with the system. It consists of a React.js[19] frontend and a Python[13] backend responsible for constructing query structures and communicating with the Solr API. Users can

choose between three search systems: the hybrid system described in Section 9.2, the semantic system described in Section 9.1, and the lexical system (labeled Keyword in the UI) based on the complex system described in Section 7.2. By default, the hybrid system is selected, as it demonstrated the best overall performance in the evaluation. However, users may switch to any of the other systems if they prefer a more customized search experience that may yield more relevant results for their specific queries. Search results are displayed in descending order of relevance, with the top entries representing the games deemed most relevant.

The interface consists of a single page. At the top, it features a search bar with the system selection options directly below it. After a search is submitted, the results appear beneath the search area, showing both the number of retrieved documents and a set of cards containing key information about each game [Figure 37]. To view more detailed information about a specific game, the user can click on its card. This action opens a modal in the center of the screen displaying all available dataset information for that game, organized into categories such as community data, communities and mechanics, creators, play details, overview information, and the full description [Figure 36].

12 Final System Characterization

After evaluating all the systems, the hybrid system was selected as the final solution. It produced the best MAP results and demonstrated the greatest flexibility, supporting both simple keyword-based queries and more expressive, detailed queries. This versatility makes it suitable for a broader and more diverse user base. For this reason, it was chosen as the default system in the user interface.

The hybrid system combines the strengths of both lexical and semantic search. On the lexical side, it applies several text-processing filters, such as lowercase normalization, ASCII folding, and stemming, to improve matching quality and reduce noise. It also incorporates synonym expansion through a synonym filter applied to the main textual fields that do not require exact matches of the original word, along with a custom stop-word filter to remove non-informative terms. These enhancements allow the system to better capture variations in user phrasing and vocabulary.

The semantic component complements this by leveraging the HNSW algorithm to compute vector-based similarity between the query context and document embeddings. This allows the system to understand meaning beyond exact word matches, especially in more descriptive or natural-language queries. By combining these two approaches, the hybrid system is able to balance precision and contextual understanding, producing more robust and reliable results across different types of information needs. The interaction between the lexical and semantic components ensures that the system can handle both structured keyword queries and richer, more expressive queries while maintaining strong performance.

13 Conclusion

This work presented the development of a board-game information retrieval system, covering data preparation, indexing, query design, and evaluation. Several retrieval strategies were implemented and compared, including basic and enhanced lexical systems, a semantic system based on vector similarity, and a hybrid system combining both approaches.

Across multiple information needs,from simple keyword searches to more expressive, descriptive queries, the evaluation showed clear differences in performance. Lexical methods excelled in precise, constraint-based queries, while semantic search was more effective for thematic or conceptual descriptions. The hybrid system consistently achieved the most balanced results, obtaining the highest MAP scores and adapting well to both structured and expressive queries.

Due to its versatility and strong performance, the hybrid system was selected as the final solution and integrated as the default option in the user interface. Its ability to combine exact term matching with contextual semantic understanding makes it particularly suitable for the diverse ways users search for board games.

The final system provides a solid foundation for future improvements, such as extending embeddings to numerical fields, enhancing query expansion, or incorporating relevance feedback to further refine retrieval quality

References

[1] 2025. Age of Majority by Country 2025. Retrieved October 12, 2025 from https://worldpopulationreview.com/country-rankings/age-of-majority-by-country
[2] 2025. all-MiniLM-L6-v2. Retrieved December 13, 2025 from https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[3] 2025. Matplotlib. Retrieved October 12, 2025 from https://matplotlib.org/
[4] 2025. NLTK. Retrieved November 15, 2025 from https://www.nltk.org/
[5] 2025. Numpy. Retrieved October 12, 2025 from https://numpy.org/
[6] 2025. Pandas. Retrieved October 12, 2025 from https://pandas.pydata.org/
[7] 2025. Scikit-learn. Retrieved October 12, 2025 from https://seaborn.pydata.org/
[8] 2025. Sentence tranformers documentation. Retrieved December 13, 2025 from https://www.sbert.net/index.html
[9] Matt Arnold. 2025. The 8 Board Games With The Longest Play Time. Retrieved October 12, 2025 from https://www.thegamer.com/board-games-longest-play-time/
[10] LLC. BoardGameGeek. 2025. Board Game Geek. Retrieved October 9, 2025 from https://boardgamegeek.com/
[11] Hinrich Schütze Christopher D. Manning, Prabhakar Raghavan. 2008. Introduction to Information Retrieval. Retrieved November 17, 2025 from https://nlp.stanford.edu/IR-book/
[12] Arthur C. Codex. 2024. Data Indexing Techniques in Apache Solr. Retrieved November 15, 2025 from https://reintech.io/blog/data-indexing-techniques-apache-solr
[13] Python Software Foundation. 2025. Python. Retrieved October 12, 2025 from https://www.python.org/
[14] The Apache Software Foundation. 2025. Dense Vector Search. Retrieved December 13, 2025 from https://solr.apache.org/guide/solr/latest/query-guide/dense-vector-search.html
[15] The Apache Software Foundation. 2025. Lucene. Retrieved November 15, 2025 from https://lucene.apache.org/
[16] The Apache Software Foundation. 2025. Solr. Retrieved November 15, 2025 from https://solr.apache.org/
[17] Google. 2025. What is semantic search? Retrieved December 12, 2025 from https://cloud.google.com/discover/what-is-semantic-search
[18] Hillary. 2023. Board Games for Large Groups. Retrieved October 12, 2025 from https://gamenightgods.com/board-games-with-high-player-counts/
[19] Inc Meta Platforms. 2025. NVIDIA CUDA. Retrieved December 13, 2025 from https://react.dev/
[20] Nvidia. 2025. NVIDIA CUDA. Retrieved December 13, 2025 from https://developer.nvidia.com/cuda
[21] Rita. 2025. Board Games for Large Groups. Retrieved October 12, 2025 from https://theboardgamecollection.com/best-board-games-for-large-groups/
[22] K Stewart. 2025. *Pearson's correlation coefficient.* Encyclopedia Britannica. Retrieved October 12, 2025 from https://www.britannica.com/topic/Pearsons-correlation-coefficient
[23] Meredith Syed. 2025. What is information retrieval? Retrieved November 15, 2025 from https://www.ibm.com/think/topics/information-retrieval
[24] Michael Waskom. 2025. Seaborn. Retrieved October 12, 2025 from https://seaborn.pydata.org/

A Annexes

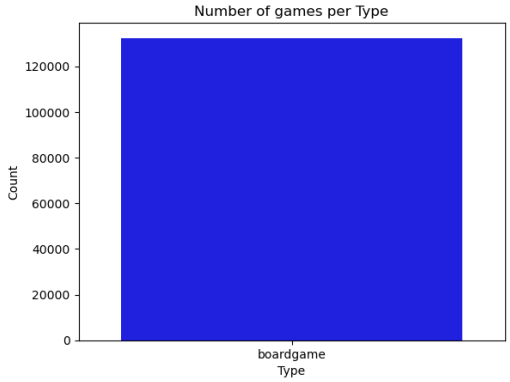

Figure 4: Games per type distribution.

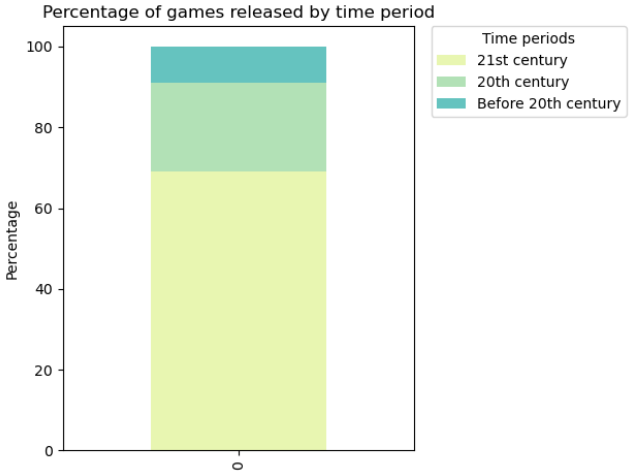

Figure 5: Release Year percentage per time period.

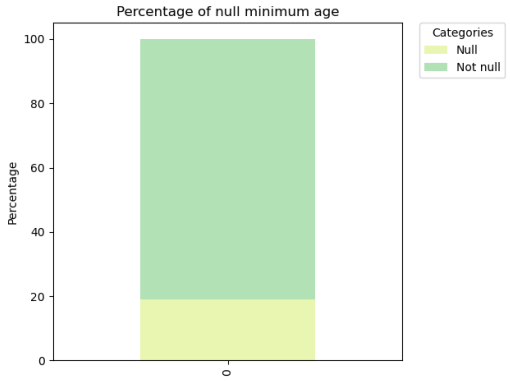

Figure 6: Percentage of nulls in the "minage" column.

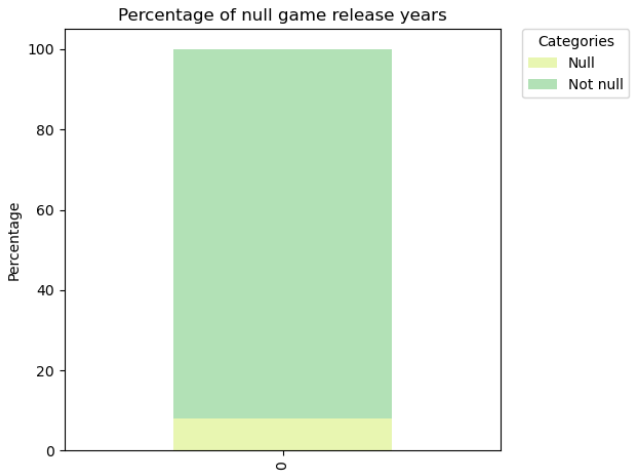

Figure 7: Percentage of nulls in the "yearPublished" column.

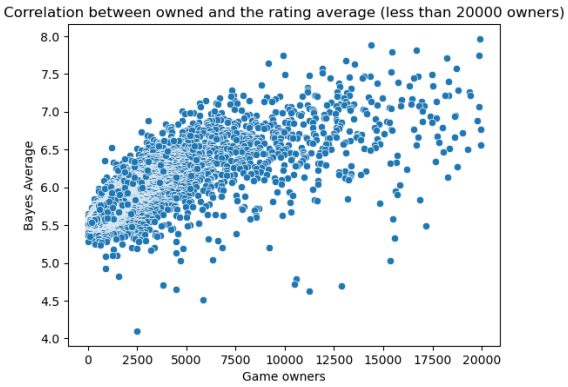

Figure 8: Correlation between the number of owners and game ratings (less than 20000 owners).

Figure 9: Correlation between the number of games in a wishlist and game ratings.

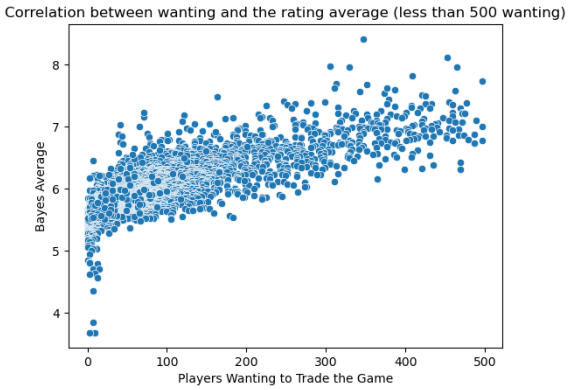

Figure 10: Correlation between the number of games users want to trade and game ratings.

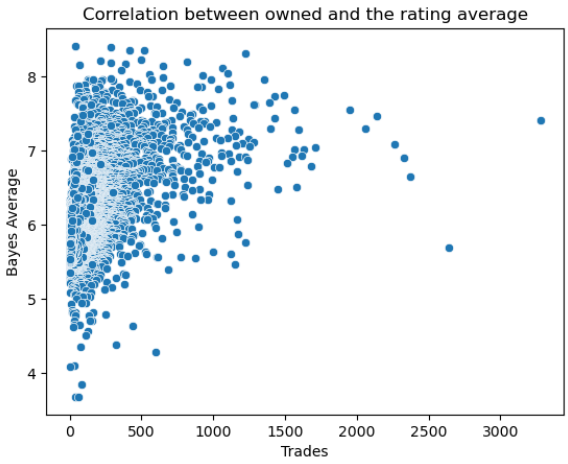

Figure 11: Correlation between the number of trades and game ratings.

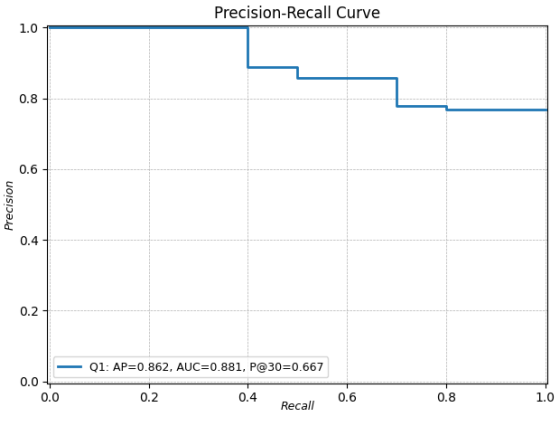

Figure 12: Precision-recall curve for Q1 (simple system).

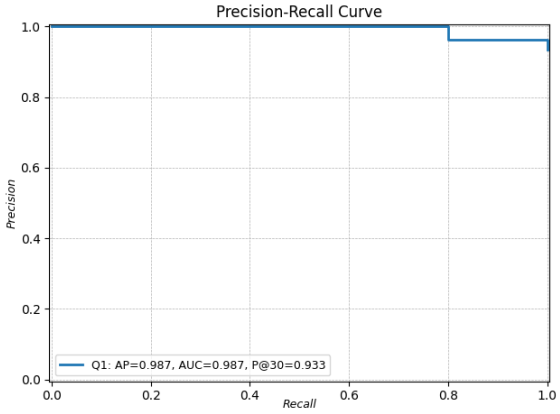

Figure 13: Precision-recall curve for Q1 (complex system).

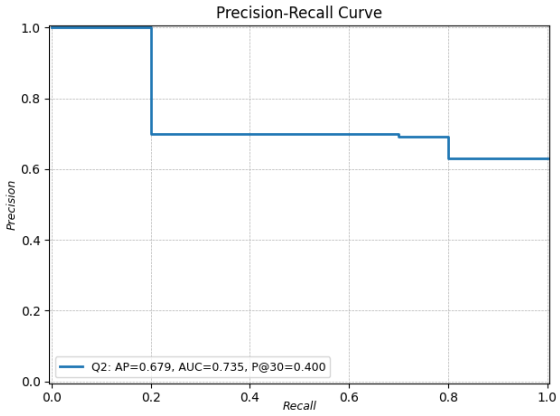

Figure 14: Precision-recall curve for Q2 (simple system).

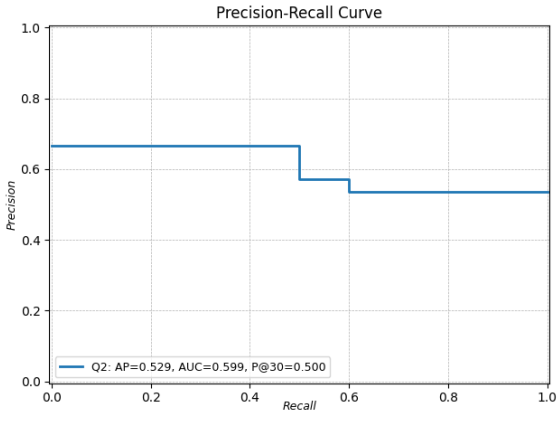

Figure 15: Precision-recall curve for Q2 (complex system).

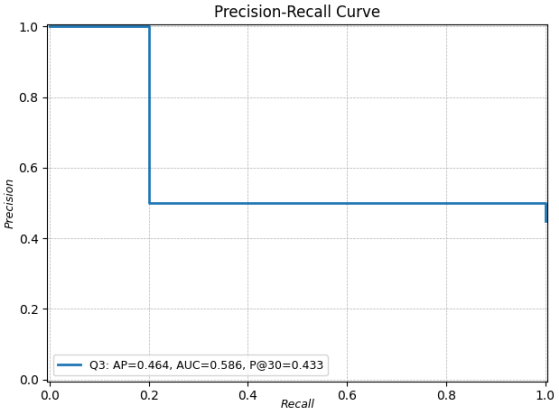

Figure 16: Precision-recall curve for Q3 (simple system).

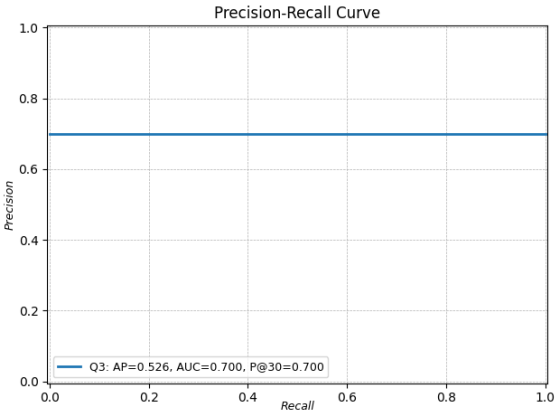

Figure 17: Precision-recall curve for Q3 (complex system).

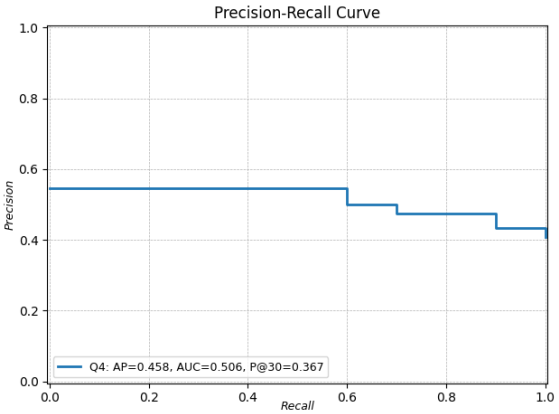

Figure 18: Precision-recall curve for Q4 (simple system).

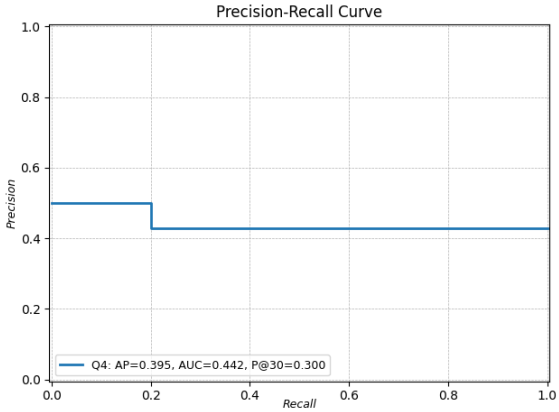

Figure 19: Precision-recall curve for Q4 (complex system).

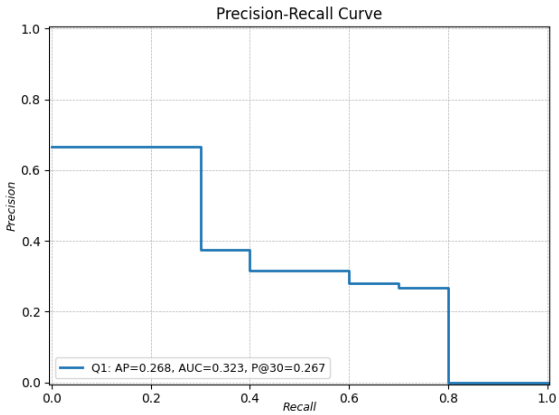

Figure 20: Precision-recall curve for Q1 (Semantic with keywords).

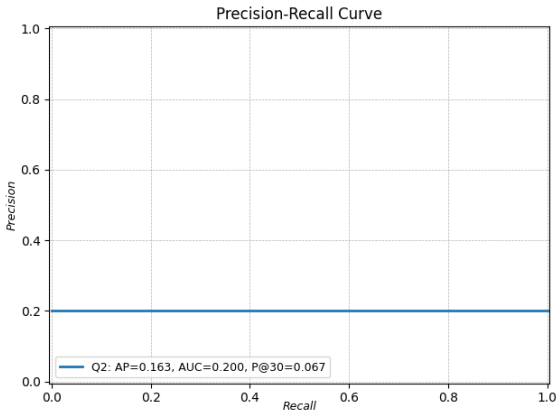

Figure 21: Precision-recall curve for Q2 (Semantic with keywords).

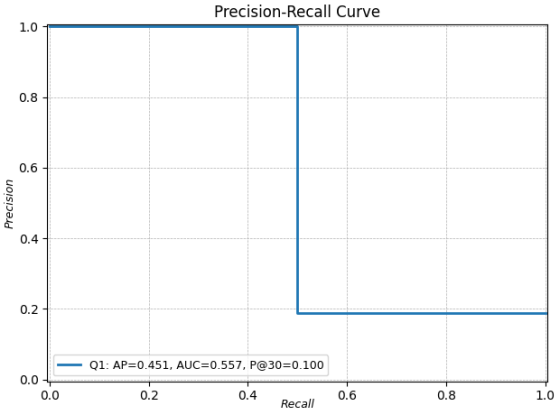

Figure 22: Precision-recall curve for Q3 (Semantic with keywords).

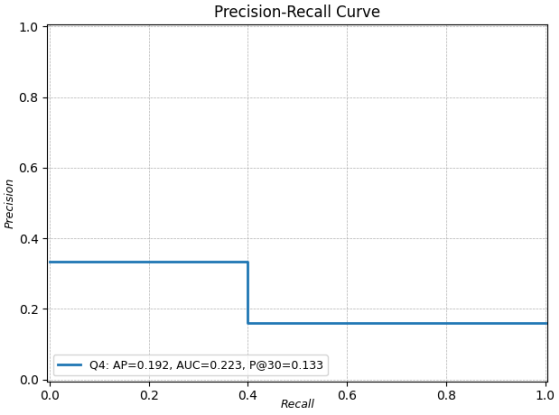

Figure 23: Precision-recall curve for Q4 (Semantic with keywords).

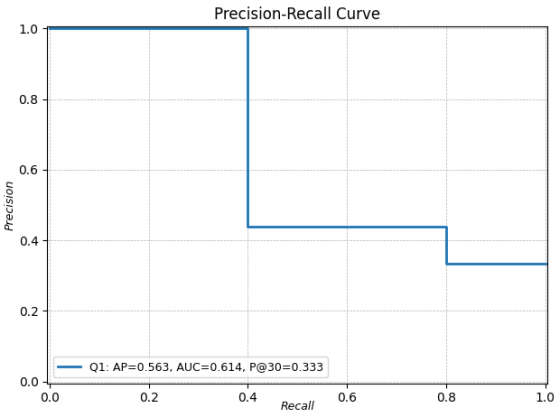

Figure 24: Precision-recall curve for Q1 (Semantic with sentences).

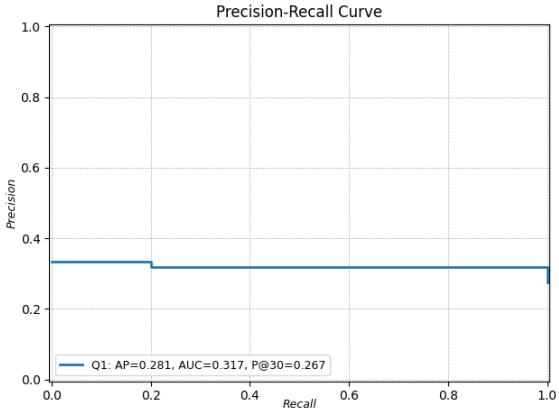

Figure 25: Precision-recall curve for Q2 (Semantic with sentences).

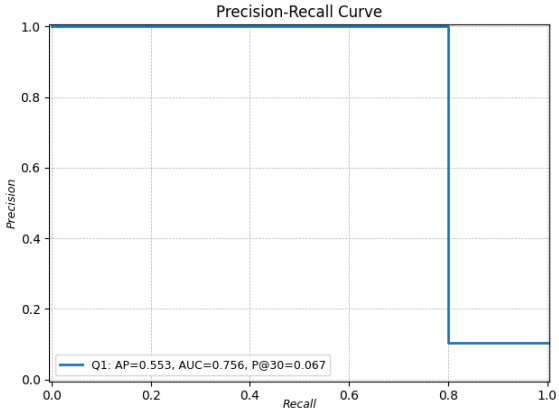

Figure 26: Precision-recall curve for Q3 (Semantic with sentences).

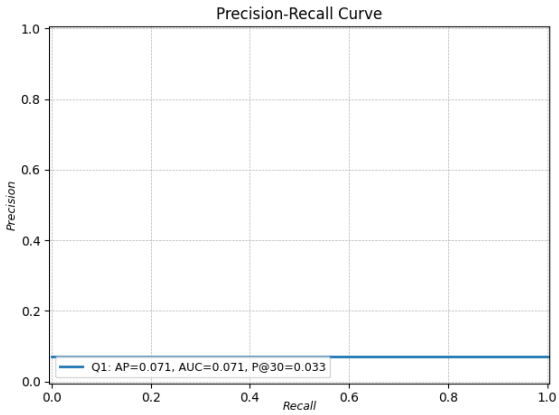

Figure 27: Precision-recall curve for Q4 (Semantic with sentences).

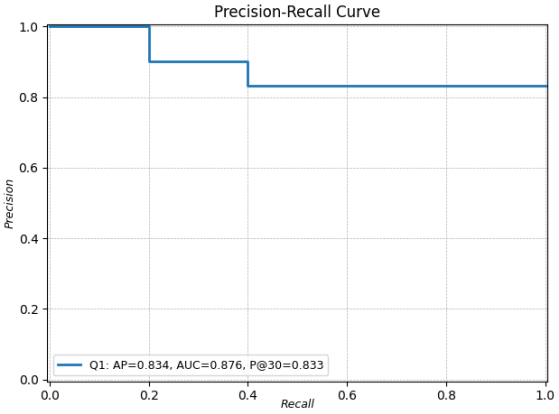

Figure 28: Precision-recall curve for Q1 (Hybrid with keywords).

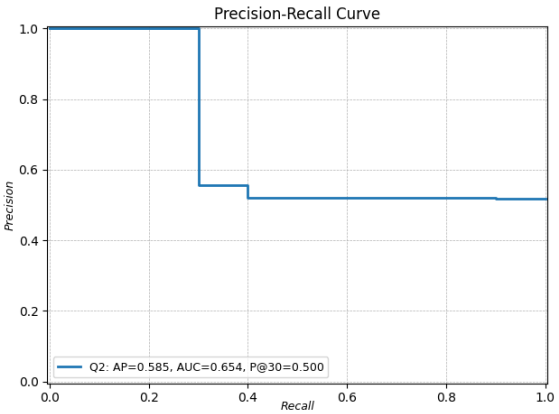

Figure 29: Precision-recall curve for Q2 (Hybrid with keywords).

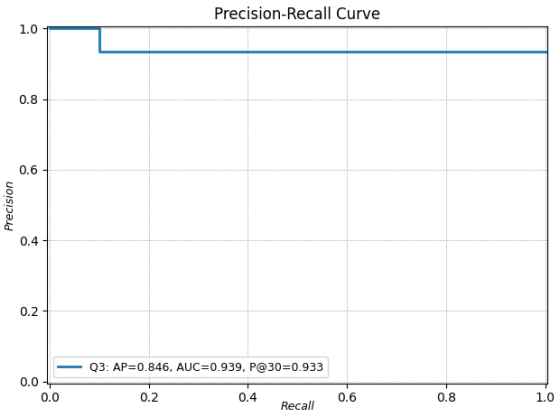

Figure 30: Precision-recall curve for Q3 (Hybrid with keywords).

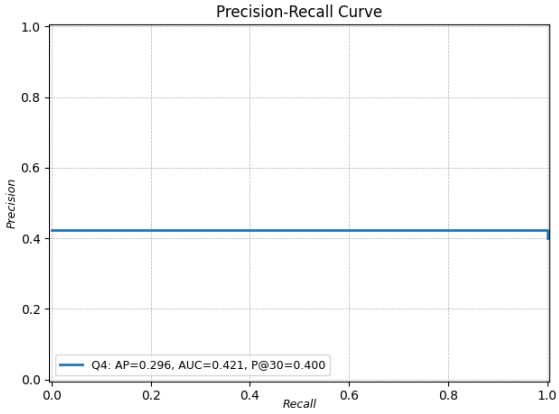

Figure 31: Precision-recall curve for Q4 (Hybrid with keywords).

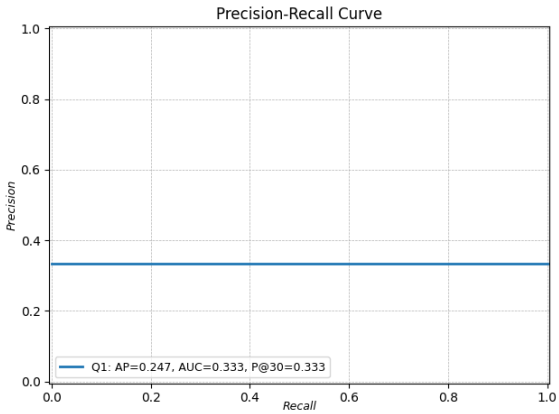

Figure 32: Precision-recall curve for Q1 (Hybrid with sentences).

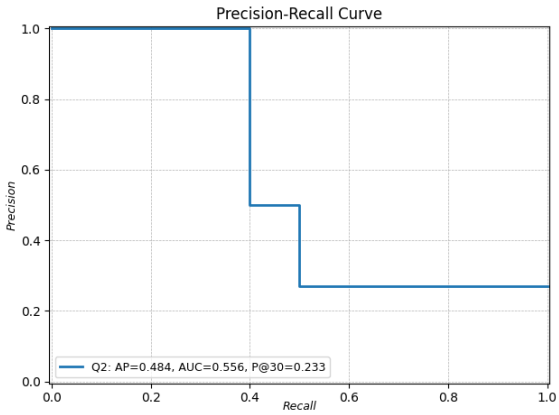

Figure 33: Precision-recall curve for Q2 (Hybrid with sentences).

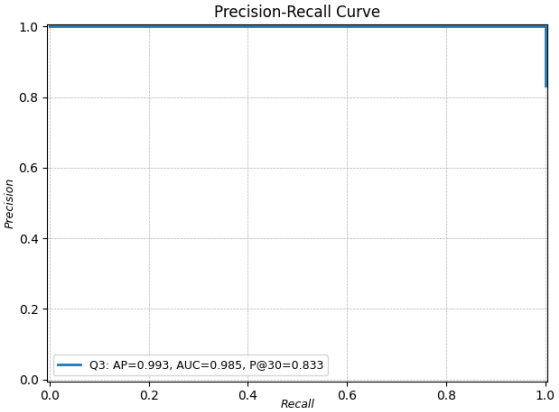

Figure 34: Precision-recall curve for Q3 (Hybrid with sentences).

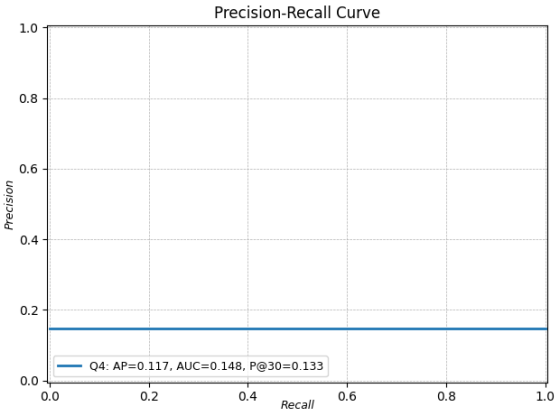

Figure 35: Precision-recall curve for Q4 (Hybrid with sentences).

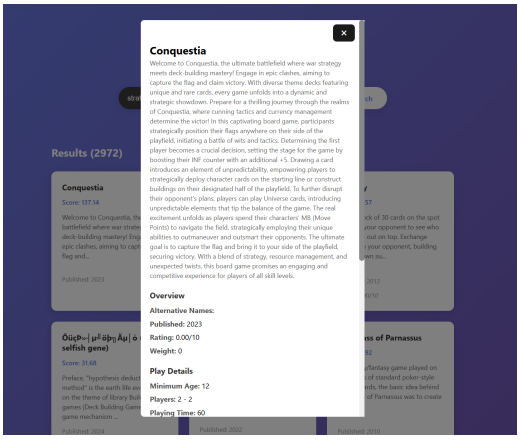

Figure 36: User interface (game information).

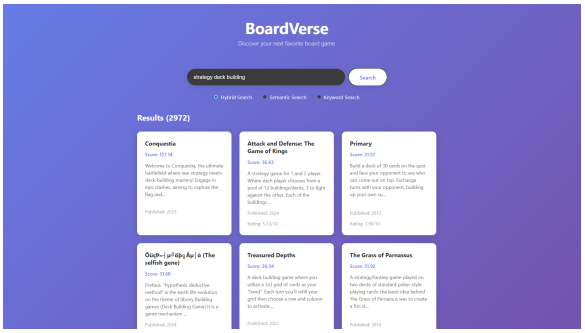

Figure 37: User interface.