Professional Videogame Reviews: Information Processing and Retrieval System

Gonçalo de Abreu Matias
Faculty of Engineering - University of Porto
up202108703@up.pt

Henrique Filipe Pereira da Silva Caridade
Faculty of Engineering - University of Porto
up202108817@up.pt

Pedro Miguel Marques da Silva Pinto
Faculty of Engineering - University of Porto
up202108826@up.pt

Abstract

With this project, we are addressing the need for an efficient system that allows users to quickly search and check reviews about potential games they might be interested in buying for example, or only searching for a second opinion about a game they played. With this article, we are documenting the methods we used to implement the Professional Video Games Reviews search engine. We used IGN's professional reviews dataset containing reviews from 1998 until today. We then used Solr for indexing and retrieval. We found some really interesting data during the course of this project, such as the fact that per year, on average 400 reviews are published, a big number considering that many people only play a few games. Another interesting fact is that scores haven't stopped increasing, thanks to the progress in the gaming industry. This project is divided into three milestones, each with different objectives. For the first Milestone 1 (Data Preparation), we started gathering data for our project. We started by creating a Python [14] script to extract the reviews for each game from IGN's website. We ended up with a CSV file, with more than 7 thousand reviews. For the second Milestone 2 (Information Retrieval), we chose Solr as our information retrieval tool. We started by indexing our documents, followed by creating queries to evaluate their results. As we only used a schema for this milestone, the results were not on par with what we really wanted. That is why for the last Milestone 3 (Search System), we improved various components of Milestone 2, such as creating a second schema, more advanced, which allowed us to get better results, or implementing the SynonymFilterFactory filter. We carried out our project by introducing semantic search with the use of embeddings and creating a modern and intuitive user interface for the search system. We can clearly say we got the highest scores and results in Milestone 3, thanks to the improvements we were able to make.

G65, October, 2024, Porto, Portugal (FEUP)

ACM ISBN 978-1-4503-XXXX-X/18/06

Keywords

Video Games, Opinions, Gamers, Professional, Reviews, Information Retrieval, Dataset, Data Collection, Data Analysis, Search Engine, Review Aggregation, Professional Criticism, User Experience, Platform Comparison

ACM Reference Format:
Gonçalo de Abreu Matias, Henrique Filipe Pereira da Silva Caridade, and Pedro Miguel Marques da Silva Pinto. 2024. Professional Videogame Reviews: Information Processing and Retrieval System. In *Proceedings of PRI (G65).* ACM, New York, NY, USA, 24 pages.

1 Introduction

In recent years, the video game industry has revolutionized the way we live and the way we play. In order for people to be able to quickly have an opinion about some game, the professional reviews, have a big influence and can even impact sales. While growing up, gaming has always been part of our lives, and that is why we think we couldn't choose any other topic for this project. We are aware it is a really rich and diverse industry, with vast amounts of data to explore, and we think it would be a nice addition and a very useful tool for people searching for gaming reviews online. That is why we chose to theme to develop this search engine for the Master's in Informatics and Computing Engineering (PRI - MEIC - FEUP). We are dividing this report into multiple sections, to provide a clear idea of what we are implementing. We start by describing why we chose this dataset, then we describe how we went about getting the dataset and the difficulties we encountered along the way. After that, we show the characteristics of the final dataset and finish by proposing some search scenarios we expect to be able to provide for our users.

2 Data Preparation

The first thing we were tasked with was to get a dataset with unstructured data and a pipeline that allowed us to extract more if need be, which we would then use in later stages of the project. This process varies wildly from topic to topic, but in our case, we didn't find any good existing datasets, so we decided to scrape from a website with game reviews.

2.1 Data Selection

After we decided that we wanted to scrape a website for our dataset we had to decide which website it would be. As there are many game review websites we searched through many of them to find one we were happy with but in the end, we landed on IGN [9]. IGN [9] is a well-known website that has vast experience with video

game reviews. It is one of the oldest in the business of video game reviews having reviews dating as far back as 1996 so after we found it, it was a no-brainer to have IGN [9] as our source of game reviews.

3 Data Pipeline

The problem with IGN [9] is that it doesn't have an API and doesn't have a place where information about every review is displayed. The only way to get the review's contents is by the review's URL, and to add insult to injury, it doesn't have a place to get all of the reviews' URLs.

The full pipeline as described in more detail ahead is illustrated in Figure 34.

3.1 Selenium Extraction

With that said, the first step in our pipeline is to get as many review URLs as possible. The best way we found to do this is with Selenium [8] because IGN's [9] game review page uses a technique known as Infinite Scrolling which means the user needs to scroll before the website loads more data, Selenium [8] is perfect for that because it allows you to mimic a user and scroll to the bottom of the page.

With Selenium [8] we just let it run for a while but found that either ign[9] or Selenium would fail at around 150th scroll and would crash the Selenium script. So we used the score filter and started by scraping the reviews with a score of 10 and the reviews with scores between 9 and 9.9 until 0. We found that this way, we could get more URLs by running it with different score ranges and 100 scrolls as the limit (most of the average score ranges got to this limit) and then end it by saving the page HTML source code to a file.

We didn't get all of the scores available on the website but we got a good part of them.

3.2 URL Extraction

The second step is to extract the review URLs from the HTML source code for all the score ranges. For that, we used BeautifulSoup [11].

BeautifulSoup [11] allows for easy traversal of the HTML document tree in Python [14]. With it, we were able to extract URLs embedded in the HTML code for every single review in it.

3.3 Review Extraction

The third and final step is to finally go through every review URL and scrape its contents. For that, we used the requests [10] library for Python [14] and BeautifulSoup [11] again.

After getting the raw HTML with the requests library we use BeautifulSoup to traverse the HTML tree and get the information we want such as the title, subtitle, author, date, the main review content, and the score.

Then using the CSV library in Python [14] we create the final product of the pipeline which is the **ign.csv** file.

4 Data Characterization

We produced numerous documents for data characterization, which helped us analyze the data more easily. We created graphs using Matplotlib [4], Seaborn [5], and Pandas [13].

4.1 Number of Reviews per Year

The first graph we created presents the Number of Reviews per Year, from 1996 until today, 2024. We can see that most reviews were published between 2007 and 2014, which correlates to the time when most video games were released to the public. Many consider this the prime of video game launches.

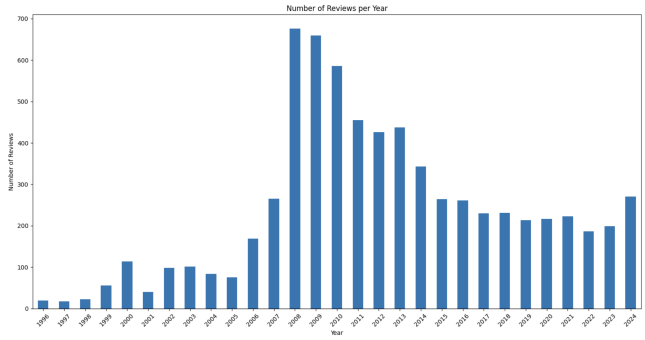

Figure 1: Reviews Per Year

4.2 Number of Reviews per Month

Figure [28] shows the number of reviews published throughout the year. We can clearly see a pattern, where the highest number of reviews publications is between the months of September and December. Normally, this is the time of the year when companies decide to launch their new video games, just before the holiday season.

4.3 Average Score per Year and Grouped Distribution of Review Scores

Figure [29] shows the score evolution over time. Back in the early two-thousands, the scores were relatively low, always in between 1 and 4. Only in 2009, we saw the first major increase in game scores. Until today, scores have not stopped increasing. We even have numerous video games maximum-rated, with a score of 10. This could be explained by the technological advancements, where developers started having the devices to produce better graphics and more immersive experiences for the players, resulting in higher scores. Another reason could be the fact that studios nowadays have way bigger budgets, so they can hire more developers and have more polished and advanced games.

Figure [30] shows almost the same data. This time around we decided to group review scores, we can clearly see the score evolution over time. We can also see the number of reviews published in those years, on top of the histograms.

4.4 Average Score by Top 20 Reviewers and Top 20 Reviewers by Number of Reviews

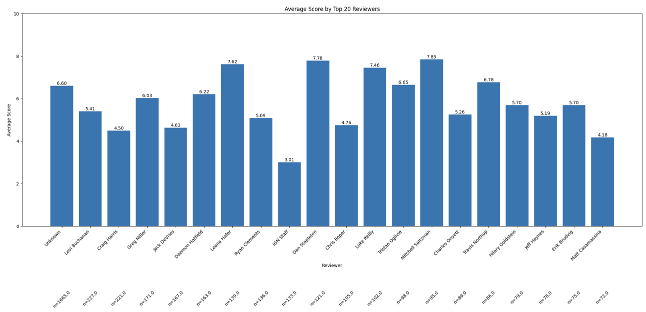

Figure 2: Average Score by Top 20 Reviewers

We also thought it would be interesting to analyze which authors tend to give higher reviews. We see that there are clearly some authors that give higher scores, maybe because some may have personal biases or preferences towards some styles or genres of games.

We also created a graph to see the author with the most published reviews (Figure [31]), and we can conclude that Levi Buchanan has 227 video games reviewed. Not far off is Craig Harris, with an astonishing 221 video games reviewed.

4.5 Word Cloud of Game Titles, Subtitles, and Content

Figure 3: Word Cloud of Game Titles, Subtitles, and Content

Word Cloud is a useful way to see which words stand out the most. In the case of the titles, we can see that the words used more often are in the first place "Review". The dataset consists of reviews, which include this term in all titles. "iPhone" because mobile games are gaining popularity, and IOS is one of the best platforms for mobile gaming. The word "Game" is fundamental and applies to all entries of the dataset.

In Figure [32], we can see that the words "Game", "New", "Fun", "Good Time", and "Best" appear regularly. This repetition shows the excitement surrounding new video game releases.

In Figure [33], we can clearly see that the word that appears the most is "Game", as the central topic of the reviews, followed by "Time", almost certainly associated with the good or bad times we can have while playing a video game, and, at last, the word "Player", that defines the person engaging with the game.

5 Search Scenarios

We identified some possible search scenarios that users would be keen to use when exploring for game reviews. These are, in our opinion, the most common ways a user would interact with our search engine:

- **Titles:** The first and most used search scenario would be to search game reviews by entering specific game titles. We think this would also be the fastest way for each user to find the review they are looking for.
- **Author:** The second possible search scenario would be to search video game reviews based on the author. It is known that some users like to find reviews from specific authors, who follow and trust their opinions.
- **Scores:** The third possible scenario would be to search for a review based on the score. A user could only be interested in games with, for example, a score above 8.
- **Subtitles:** The fourth search scenario would be to search for reviews based on their subtitles, which sometimes contain keywords like we saw thanks to the Word Cloud analysis.
- **Date:** The fifth search scenario consists of searching for a review based on the publish date. A user with a gaming machine from the latest generation would only be interested in finding reviews for games compatible with his device, and not searching for game reviews of games some years old.
- **Genre:** The sixth and last search scenario we think would be realistic is to search for a video game review based on the genres. Normally people tend to like specific types of games, so allowing the user to search by genre would be a very welcome functionality.
- **Games with tight controls:** Responsive controls are very important for gamers. Having input lag is one of the biggest issues in the gaming world. Gamers want precise and responsive mechanics. That is why we chose this search scenario as a possible one.
- **Games good for relaxing:** People interested in relaxing games, that don't want the stress of competitive games, could be keen to search for these types of games. These are the perfect games to play after a long and tiring day.
- **Story narrative games:** Some players would be keen to search for good story narrative games, a style of game really appreciated by many gamers.
- **Technical aspects of a game:** Graphics is a really important topic for gamers, so we imagine many would be interested in searching for a technical performance of a certain game in a certain platform, such as frame rates, loading times, or resolution quality.
- **Social gaming experience:** Multiplayer is a key feature in today's games. This is what keeps gamers interested in games after finishing the single-player mode. A typical gamer would search for a reliable multiplayer experience, where they can play seamlessly with their friends.

6 Information Retrieval

We now begin the second part of this project: Information retrieval, which involves extracting relevant information from large datasets. This process focuses on searching and retrieving specific information from documents. We will now explain and present each method we used in this part of the project.

7 Collection and Indexing

7.1 Document Definition

The game reviews we extracted from IGN's website serve as the primary documents. We converted our CSV file to a JSON [1] file, and we added the **Id** attribute to each game review. We then created a subset file, to better improve query search, then marking **Id** as relevant or not.

Each document contains:

- **Title:** Game name and review indicator
- **Subtitle:** Brief tagline
- **Subheader:** Author and Date
- **Content:** Full review text
- **Score:** Numerical rating (0-10)
- **Id:** Individual identification for each review

In regard to the configuration, we used Solr [6] (a search platform from Apache), because it provides text search and real-time indexing capabilities for large document collections, features that we think are essential to handle our large dataset.

The following configurations were implemented for optimal document processing:

- **Text Fields (Title, Content, Subtitle)**
  - Field Type: `game_text`
  - Processing:
    - Standard Tokenization
    - Lowercase Filtering
    - Stop Word Removal
  - Purpose: Enables flexible and advanced text searching and analysis, supporting complex tokenization and stemming
- **Score Field**
  - Field Type: `pfloat`
  - Capability: Supports range queries, numeric operations, and score-based filtering
  - Purpose: Enables score-based filtering and analysis

7.2 Indexing Process

This was one of the most important parts of our project because it allows fast search and retrieval, as well as relevant ranking, field-specific queries, complex queries, and full-text search.

7.3 Schema Details

We used a schema to index our documents for the search system and so we decided to build a simple schema that is adapted for our retrieval tasks. For example, the *text_simple* as a custom field was designed for basic text processing. *pfloat* was used for decimal numbers, more precisely for the score field. The **StandardTokenizer** is used to split text into words. This way, it makes text searchable by individual words. **LowerCaseFilter** converts a text to lowercase, so that we get all text to the same case, and so we don't have issues with for example "games" and "Games": both results should be retrieved. **StopFilter** removes words not important, like "the", "and", "is", reducing the index size. All fields were indexed to improve search performance, and all fields were stored so that the original value could be retrieved.

Field	Type	Indexed
Title	text_simple	Yes
Content	text_simple	Yes
Subtitle	text_simple	Yes
Subheader	text_simple	Yes
Score	pfloat	Yes

Table 1: Overview of Important Fields in the Schema

- **Main title of the document** — Corresponds to the field `Title`
- **Full body content of the document** — Corresponds to the field `Content`
- **Subtitle providing additional context** — Corresponds to the field `Subtitle`
- **Secondary header for organizing content** — Corresponds to the field `Subheader`
- **Numeric score used for ranking and sorting** — Corresponds to the field `Score`

8 Retrieval Process

For this part of the project, we used our simple schema.

The following table describes the key query parameters used in Solr [7]:

(1) **Query (q)**
Focuses on the most valuable words in the query.
(2) **Query Operator (q.op)**
Utilizes OR or AND for query operations.
(3) **Query Filter (fq)**
Defines a query to restrict the superset of documents.
(4) **Filter List (fl)**
Limits the fields included in a query response.
(5) **Sort Field (sort)**
Specifies the sorting value for the results.
(6) **Start (start)**
Specifies the starting offset for paginated results.
(7) **Rows (rows)**
Limits the number of documents returned in a single query.
(8) **Query Parser (defType)**
Specifies the query parser (e.g., edismax).
(9) **Boost Query (bq)**
Adds boosting to specific query terms to increase their relevance.
(10) **Highlighting (hl)**
Enables highlighting of matched terms in the results.

The query parsers are tools that interpret and process user queries. They essentially convert text to a query format that is then used in the search engine. Queries support boolean operators like **AND, OR, NOT**. They also allow for title search for example. On the other hand, **Dismax** is designed for simple keyword

searches, and it searches fields like title and content. It also allows for boosting. There is an improved version of **Dismax**, called **eDismax** [3], that supports more advanced query syntax and allows for example field-specific boosts.

9 Evaluation Process

Another important part of this project was the evaluation process. We can now identify user search scenarios, clarify types of documents, and specify the goals of the evaluation process. We used different metrics, such as **Precision-Recall Curves**, **P@**, **MAP** and **AUC**. **Precision Recall Curves** allows us to visualize the trade-off between precision and recall, so, it's effective when we want to check the relevance of retrieved items and the coverage of all relevant items. This way, we ensure a balanced performance for both precision and recall. On the other hand, **P@** is useful to evaluate the precision of retrieved results, in other words, it shows the quality of the top results. The **Mean Average Precision (MAP)**, essentially summarizes precision across queries, and gives more weight to relevant items that were retrieved. Thanks to this, we can see the relevant results across the different queries we implemented during our evaluation process. The **Area Under the Curve (AUC)**, helps us see the performance across all possible classification thresholds. This is how we know if the system can distinguish between relevant and non-relevant items. We think these metrics provide a well-rounded evaluation for our retrieval system.

We will now present each evaluation we performed using these metrics.

9.1 Games with tight controls

In the first evaluation process, we decided to find games with tight controls, so, games with a high degree of responsiveness and precision. This way, players feel a strong sense of control, which results in a natural and engaging game-play. The gamer knows he is in control of everything.

We used keywords like **controls**, **precision**, **responsive**, **accurate**. We think these are the most important words to search for in this particular case, that would lead us to the best results.

Table 2: Query Simple

Query	
q	`Content:(*tight* *responsive* *precise*) OR Subtitle:(*tight* *responsive* *precise*)`
qf	`Content^4.0 Subtitle^2.0`
defType	`edismax`
rows	`100`
fl	`id,Title,Content,Score`

Table 3: Query Boosted

Query	
q	`(Title:(*controls* *responsive* *precise* *tight*) OR Content:(*controls* *responsive* *precise* *tight* *accurate* *fluid* *precision*) OR Subtitle:(*controls* *responsive* *precise* *tight*))`
qf	`Title^2.0 Content^3.0 Subtitle^2.5`
pf	`Title^6.0 Content^3.0`
bq	`Subtitle:(*precise*)^5.0 Content:(*fluid* *responsive*)^3.0`
fl	`id,Title,Content,Score`

Table 4: Evaluation Results

Evaluation Metrics	
AVP	0.873
MAP	0.457
AUC	0.493
P@5	1.00
P@10	0.90
P@15	0.80
P@20	0.85

We can see that the boosted query got better results. It used more specific terms, so we think that was key to better target content. The OR condition allowed the boosted query to capture way more content.

9.2 Games good for relaxing

For the second evaluation process, we decided to find games good for relaxing, so, the perfect games to play after a long day. These games focus on minimal challenges, calm game-plays, and smooth mechanics.

We used keywords like **relaxing**, **peaceful**, **calming**, **chill**. We think these are the most important words to search for in this particular case that would lead us to the best results.

Table 5: Query Simple

Query	
q	`Title:(*relax* *peaceful* *calm* *chill*) OR Content:(*relax* *peaceful* *calm* *meditate* *chill* *simple* *serene* *enjoy* *soft*) OR Subtitle:(*relax* *peaceful* *calm* *meditate* *chill*)`
qf	`Title^2.0 Content^3.0 Subtitle^2.5`
defType	`edismax`
bf	`Score^2.0`
rows	`100`
fl	`id,Title,Score`

Table 6: Query Boosted

Query	
q	`Title:(*relax* *peaceful* *calm* *chill*) OR Content:(*relax* *peaceful* *calm* *meditate* *chill* *mindful* *simple* *serene* *enjoy* *soft*)`
qf	`Title^4.0 Content^2.0 Subtitle^1.5`
pf	`Title^8.0 Content^4.0`
bq	`Subtitle:(*peaceful*)^2.0 Content(*calm* *relax*)^3.0`
fl	`id,Title,Score`

Table 7: Evaluation Results

Evaluation Metrics	
AVP	0.354
MAP	0.172
AUC	0.178
P@5	0.60
P@10	0.40
P@15	0.33
P@20	0.30

Again, we got better results with the boosted query. It used more specific terms to better target games that focus on relaxation and calmness. The OR condition allows one more time to capture way more relevant content.

9.3 Story narrative games

For the third evaluation process, we decided to find story narrative games, games that emphasize strong storytelling, character development, and immersive plots. There are games where player choice is prioritized.

We used keywords like **story**, **narrative**, **emotional**, **character**. We think these are the most important words to search for in this particular case that would lead us to the best results.

Table 8: Query Simple

Query	
q	`Content:(*story* *narrative*) OR Subtitle:(*story* *narrative*)`
qf	`Content^4.0 Subtitle^2.0`
defType	`edismax`
rows	`100`
fl	`id,Title,Content,Score`

Table 9: Query Boosted

Query	
q	`Title:(*story* *narrative* *emotional* *character*) OR Content:(*story* *narrative* *emotional* *character*) OR Subtitle:(*story* *narrative* *emotional* *character*)`
qf	`Title^2.0 Content^3.0 Subtitle^2.5`
pf	`Title^6.0 Content^3.0`
bq	`Subtitle:(*story*)^5.0 Content:(*character* *emotional*)^4.0`
fl	`id,Title,Content,Score`

Table 10: Evaluation Results

Evaluation Metrics	
AVP	0.748
MAP	0.491
AUC	0.506
P@5	0.80
P@10	0.90
P@15	0.80
P@20	0.75

This time around, we got better results for the simple query. The boosted query may have had a way more complicated structure, which led to finding worse results. The simple query targeted keyword selection, so highly specific words. These terms probably were enough to capture the elements we were looking for. The boosted query additional terms may have introduced unnecessary complexity.

9.4 Visually impressive Games

For the fourth evaluation process, we decided to find visually impressive games, so, games that stand out due to their graphics and visual design. There are games that often showcase cutting-edge technology, that offer stunning details for the player.

We used keywords like **visually**, **graphically**, **visuals**, **realistic**. We think these are the most important words to search for in this particular case that would lead us to the best results.

Table 11: Query Simple

Query	
q	`Content:(*graphically* *visuals*) OR Subtitle:(*graphically* *visuals*)`
qf	`Content^4.0 Subtitle^2.0`
defType	`edismax`
rows	`100`
fl	`id,Title,Content,Score`

Table 12: Query Boosted

Query	
q	`Title:(*visually* *graphically* *visuals* *realistic*) OR Content:(*visually* *graphically* *visuals* *realistic*) OR Subtitle:(*visually* *graphically* *visuals* *realistic*)`
qf	`Title^2.0 Content^3.0 Subtitle^2.5`
pf	`Title^6.0 Content^3.0`
bq	`Subtitle:(*visuals*)^4.0 Content:(*realistic*)^3.0`
fl	`id,Title,Content,Score`

Table 13: Evaluation Results

Evaluation Metrics	
AVP	0.353
MAP	0.090
AUC	0.096
P@5	0.40
P@10	0.30
P@15	0.33
P@20	0.35

We got really low results in both simple and boosted queries. We think this might due to the fact that both queries didn't find relevant content, because **review authors** maybe don't focus as much on **technical aspects** of the game, and so it was difficult for queries to find good results.

9.5 Multiplayer Games

For the fifth and last evaluation process, we decided to search for Multiplayer Games. These are games that allow multiple players to play together in the same game. Normally these games offer competitive modes, that make them ideal for players with friends or family.

We used keywords like **online**, **cooperative**, **competitive**, **team**. We think these are the most important words to search for in this particular case that would lead us to the best results.

Table 14: Query Simple

Query	
q	`Title:(*multiplayer* *online*) OR Content:(*multiplayer*)`
qf	`Title^3.0 Content^2.0`
defType	`edismax`
rows	`100`
fl	`id,Title,Content,Score`

Table 15: Query Boosted

Query	
q	`Title:(*multiplayer* *online*) OR Content: (*multiplayer* *team*) Subtitle: (*multiplayer* *online* *cooperative* *team*)`
qf	`Title^4.0 Content^2.0 Subtitle^1.5`
fq	`Content:multiplayer`
bq	`Subtitle:(*online*)^3.0 Content:(*multiplayer*)^5.0`
fl	`id,Title,Content,Score`

Table 16: Evaluation Results

Evaluation Metrics	
AVP	0.781
MAP	0.308
AUC	0.303
P@5	0.60
P@10	0.80
P@15	0.80
P@20	0.85

In these two last queries, we got similar results between simple and boosted queries. Both queries used the core "**Multiplayer**" term, so maybe both queries found similar results. Searching for this type of game is straightforward, so maybe a simple query can get the job done, and in this case, both queries did well.

Now that Milestone 1 and 2 have finished, we are excited to move on to Milestone 3. In Milestone 2, we collected and indexed documents, performed retrieval tasks, and evaluated the different results we obtained from the queries. We are satisfied with the progress we have made so far in these two Milestones, and we think this latest Milestone is a solid foundation for the future, and that is why we can't wait for Milestone 3 to start, to finally implement some improvements. We will try to implement a second, more advanced schema, and so have two schemas: a **simple** and a **more enhanced** one. We also need to start implementing some **query expansion strategies**, for example, to find synonyms and related terms, in order to try to improve the results of our queries.

10 Search System

We now start Milestone 3 (Search System) with **many improvements** over the previous Milestone. Our objective was to improve query results.

10.1 Improvements

Boosted Schema

Apart from the original single schema we had in the second Milestone, we now have a second one, **more advanced**, with even **more filters**. To evaluate relevant and non-relevant documents, we created two queries. The first one, simpler, runs the simple schema.

The second, more advanced, runs the boosted schema. Then, a JSON [1] file is created with the documents. We then manually mark the results as relevant or not, to then be used in our evaluation of the system.

Table 17: Boosted Schema Analysis Chain

Analysis Chain	
StandardTokenizerFactory	Splits text into tokens using word boundaries
LowerCaseFilterFactory	Converts all text to lowercase
StopFilterFactory	Removes common stop words
EnglishPossessiveFilterFactory	Removes possessive endings ('s) from words
EnglishMinimalStemFilterFactory	Performs minimal stemming on English words
SynonymGraphFilterFactory	Handles synonym expansion with: - synonyms: configured file path - expand: true - ignoreCase: true

Table 18: Simple Schema Analysis Chain

Basic Analysis Chain	
StandardTokenizerFactory	Splits text into tokens using word boundaries
Filters	None applied
Additional Processing	None configured

SynonymFilterFactory filter

We implemented the **SynonymFilterFactory filter**. This way, there is no need to use all the synonyms in the queries. We created a synonyms.txt file, where we store the synonyms for the **most important words** for each previous **search scenarios** we created (Games with tight controls, Games good for relaxing, Story narrative games, Visually Impressive Games, and Multiplayer Games).

Table 19: Search Scenarios and some of their Synonyms

Search Scenarios	
Games with Tight Controls	handling, inputs, gameplay, maneuverability
Games Good for Relaxing	calm, peaceful, zen, tranquil
Story Narrative Games	cutscenes, moments, dialogue, emotional
Visually Impressive Games	fidelity, rendering, animation, quality
Multiplayer Games	crossplay, teams, friends, group

Thanks to these new improvements, we can clearly now see that a **more advanced schema** can really give **better results** than the simple schema. This was not the case in our second Milestone, we were using the same schema for simple and boosted queries, so there was no real improvement between each system.

Games with Tight Controls

Table 20: Query Simple

Query	
q	`Content:(controls good tight precise responsive fluid excellent smooth)`
rows	`30`
fl	`id,Title,Score,Subtitle,Content`
sort	`score desc`

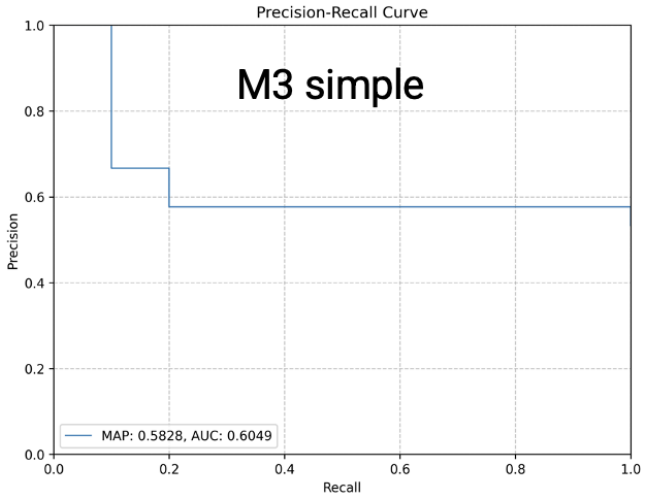

Figure 4: Games with tight controls simple query

Table 21: Query Boosted

Query	
q	`Content:(controls responsive tight precise fluid gameplay handling)^4 OR Content:(gameplay movement physics)^2 OR Title:(controls gameplay)^3`
defType	`edismax`
qf	`Content^3 Title^2 Subtitle`
pf	`Content^10`
mm	`2<70%`
rows	`30`
fl	`id,Title,Score,Subtitle,Content`

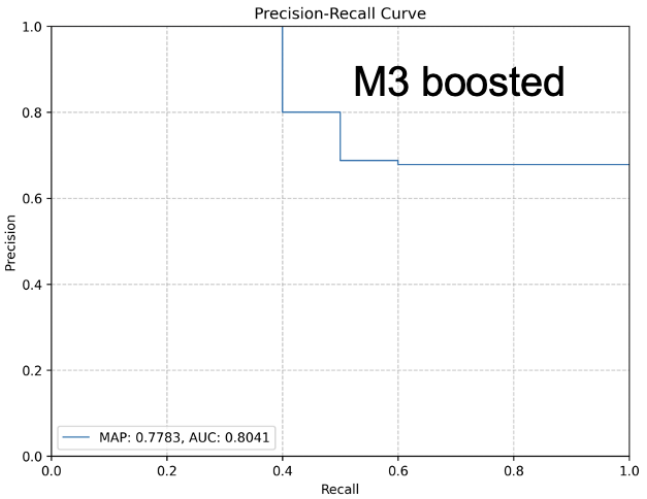

Figure 5: Games with tight controls boosted query

We can clearly see the **more advanced schema** gets way **better results**, thanks to the **powerful filters** we managed to implement, such as the Synonym Filter.

Visually Impressive Games

We can also compare the performance between Milestone 2 queries using the same schema and Milestone 3 queries using two different schemas. We decided to compare the results for **Visually Impressive Games**, as the improvement is notorious.

Table 22: Query Simple

Query	
q	`Content:(graphics visuals performance technical resolution framerate)`
rows	`30`
fl	`id,Title,Score,Subtitle,Content`
sort	`score desc`

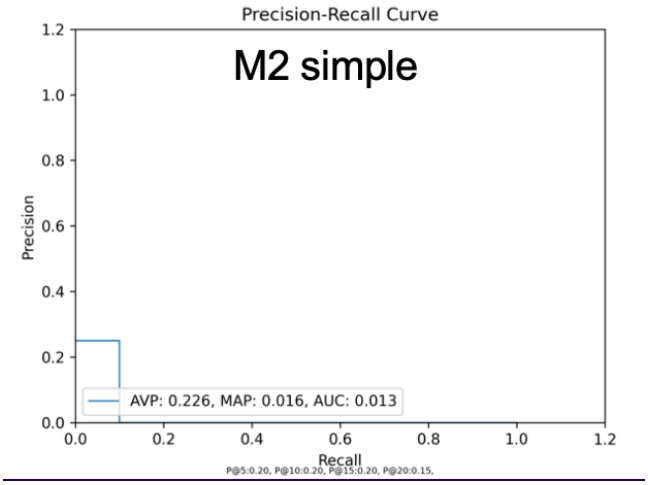

Figure 6: Visually Impressive Games simple query

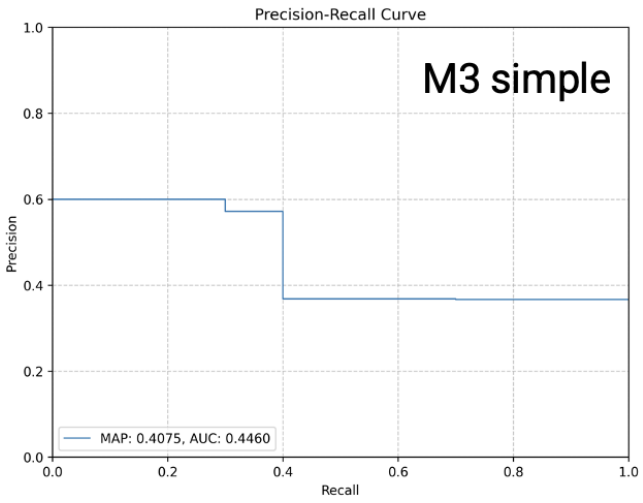

Figure 7: Visually Impressive Games simple query

Table 23: Query Boosted

Query	
q	`Content:(graphics visuals fidelity ray-tracing performance)^4 OR Content:(optimization framerate resolution rendered)^3 OR Content:(textures lighting effects animations)^2 OR Title:(graphics performance technical)^3`
defType	`edismax`
qf	`Content^3 Title^2 Subtitle`
pf	`Content^10`
mm	`2<70%`
bf	`Score^2`
rows	`30`
fl	`id,Title,Score,Subtitle,Content`

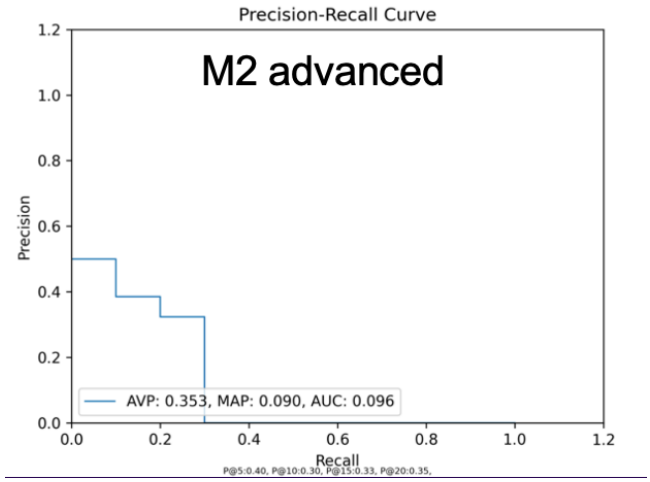

Figure 8: Visually Impressive Games advanced query

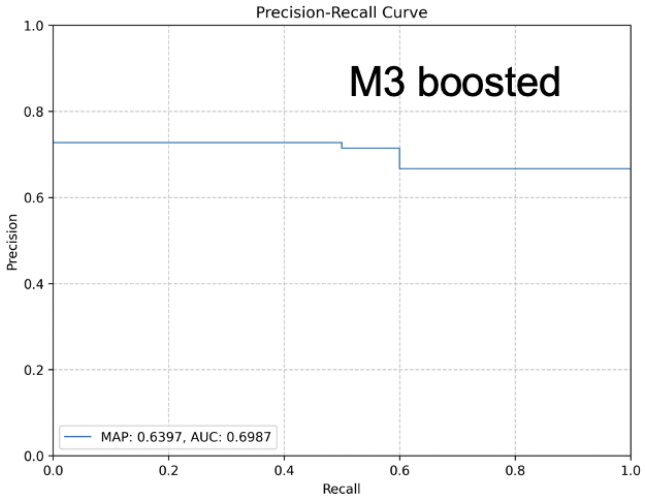

Figure 9: Visually Impressive Games boosted query

We can clearly see a **huge improvement** here. The new filters, and having a **more advanced schema** with more powerful features, really help the system retrieve more relevant results. In Milestone 2, for the simple query, the results were very poor. There was a **huge drop** in precision, as the score barely reached 0,2. It was a very limited system for finding the games we wanted. On the other hand, we got a precision of around 0,6 for the Milestone 3 simple query, even if it drops after. It's already an improvement over the previous milestone. For the **advanced query**, still in Milestone 2, there was an improvement, width a **higher precision** (around 0.4), and a **slightly better performance** due to **boosting**. But, with the boosted schema we created, we even got **better results,** with a much more **stable performance curve**, that maintains a high precision, around 0,7. We are sure the implementation of the **Synonym Filter** also played a vital role in getting these **better results**, as more related words could be found that were related to a certain topic. These improvements really show the importance of having proper text and filter analysis in our schemas.

More Precision-Recall Curves are available in the attachment section.

10.2 Semantic Search

For Milestone 3 we decided to implement semantic search because, for some search scenarios, the results we were getting were **not high enough**. Relaxing and Visually Impressive Games had really low results when compared to the rest. Semantic Search allows the system to understand the context behind our queries. This method allows the search system to find documents that are semantically related to the queries we created. The nearest neighbor search system algorithm is used to identify the closest dense vectors for each query.

We started by generating document embeddings with the help of a python [14] script, that uses the **sentence-transformers** library. We chose the **all-MiniLM-L6-v2** model, as this was recommended by the teachers. We used our original JSON [1] file as input for the script. We also created a **new core** and applied to it a **new semantic schema**, where we incorporated a new field type for the **DenseVectorField**. Our final JSON [1] file includes the same fields as the original one, but the reviews are divided into paragraphs. This way, it is easier to find relevant parts of the reviews that match our queries.

Below are the results we got.

Games with Tight Controls

We decided to first evaluate **Games with Tight controls**. These are games that are **responsive**, **precise**, have an **engaging gameplay** and where the user feels he is connected to the game and the game responds well to the user inputs.

The query terms are first converted to dense vector embeddings using the **SentenceTransformer** model, and these vectors are used for the actual semantic search. The results were then manually evaluated for relevance to generate precision-recall metrics.

Table 24: Semantic Search Query

Semantic Search Query	
q	`{!knn f=vector topK=50}controls responsive precise`
Description	Uses dense vectors generated by all-MiniLM-L6-v2 model to find semantically similar content. The query text is converted to a 384-dimensional vector before being sent to Solr.
fl	`id,parent_id,Title,Content,Score, paragraph_num,score`
rows	`30`
wt	`json`
sort	`score desc`

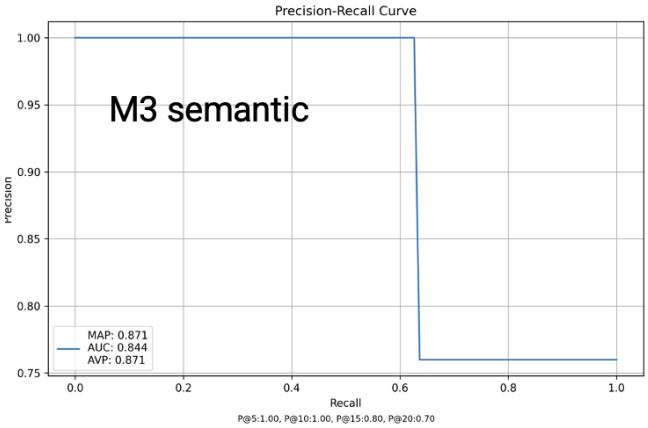

Figure 10: Games with tight controls semantic query

Table 25: M3 Comparison

Rank	Simple	Boosted	Semantic
1	R	R	R
2	N	R	R
3	R	R	R
4	N	R	R
5	R	R	R
6	N	R	R
7	R	N	R
8	N	N	R
9	R	R	R
10	N	R	R
P@10	0.50	0.80	1
AvP	0.6787	0.9472	1

For this **semantic search**, we use three key terms: "controls, responsive and precise". We search using the topK=50 so that we can get a high coverage of results. We got a really high MAP and precision score, showing a really robust overall performance. We can say that we get very few **false positives** because we have a **high precision**. This clearly shows that the semantic search is searching beyond just simple keyword matching.

Multiplayer Games

For **multiplayer games**, we are searching for games that allow users to play in groups or teams, with or without friends. We expect good results because thanks to the synonyms we implemented, it is really easy to find relevant multiplayer games.

Table 26: Semantic Search Query

Semantic Search Query	
q	`{!knn f=vector topK=50}multiplayer online cooperative`
Description	Uses dense vectors generated by all-MiniLM-L6-v2 model to find semantically similar content. The query text is converted to a 384-dimensional vector before being sent to Solr.
fl	`id,parent_id,Title,Content,Score, paragraph_num,score`
rows	`30`
wt	`json`
sort	`score desc`

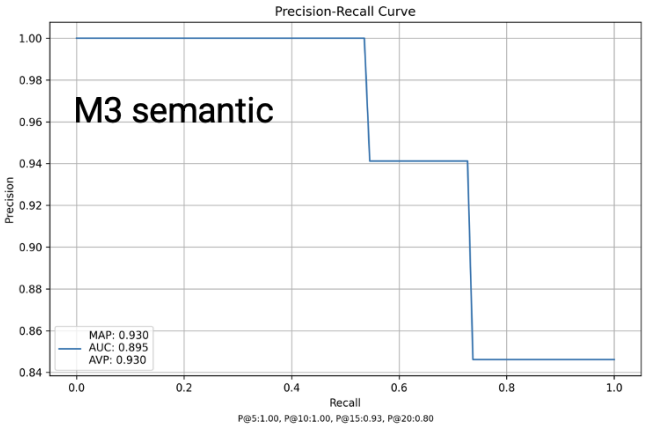

Figure 11: Multiplayer Games semantic query

Table 27: M3 Comparison

Rank	Simple	Boosted	Semantic
1	R	R	R
2	N	R	R
3	N	R	R
4	N	R	R
5	R	N	R
6	R	N	R
7	R	N	R
8	N	R	R
9	R	R	R
10	N	R	R
P@10	0.50	0.70	1
AvP	0.6054	0.8560	1

As expected, with got excellent results. It was really easy for the system to find multiplayer game reviews.

Story Narrative Games

We now want to search for **games with heavy story-telling**, where the user is involved and its decisions have an impact on the story of the game. These are games with **strong dialog and emotional stories**.

Table 28: Semantic Search Query

Semantic Search Query	
q	`{!knn f=vector topK=50}story narrative`
Description	Uses dense vectors generated by all-MiniLM-L6-v2 model to find semantically similar content. The query text is converted to a 384-dimensional vector before being sent to Solr.
fl	`id,parent_id,Title,Content,Score, paragraph_num,score`
rows	`30`
wt	`json`
sort	`score desc`

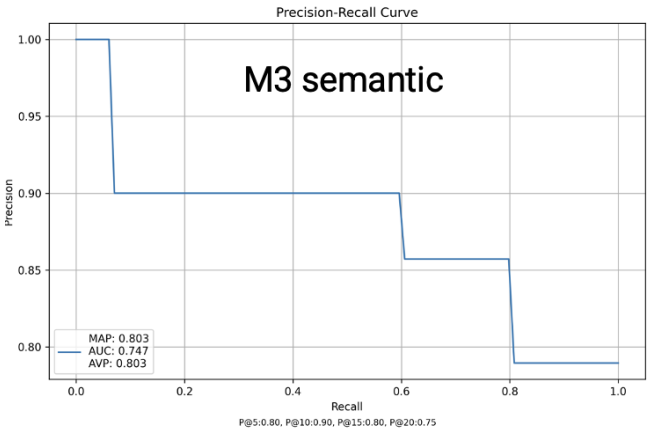

Figure 12: Story Narrative Games semantic query

Table 29: M3 Comparison

Rank	Simple	Boosted	Semantic
1	N	N	R
2	R	N	N
3	R	R	R
4	R	N	R
5	N	N	R
6	R	R	R
7	R	R	R
8	R	R	R
9	R	R	R
10	R	R	R
P@10	0.80	0.60	0.90
AvP	0.7032	0.4585	0.8412

We got **really good results** this time. Even though we got an early drop in recall, the system maintained its precision quite well.

Even when retrieving more results, the system maintains **strong precision**.

Visually Impressive Games

This was one of the search scenarios we wanted to improve heading into Milestone 3 because previous Milestone 2 results were fairly low. We are searching for games that can provide a splendid experience to the user, with **very good visuals and graphics.**

Table 30: Semantic Search Query

Semantic Search Query	
q	`{!knn f=vector topK=50}graphics visuals performance technical`
Description	Uses dense vectors generated by all-MiniLM-L6-v2 model to find semantically similar content. The query text is converted to a 384-dimensional vector before being sent to Solr.
fl	`id,parent_id,Title,Content,Score, paragraph_num,score`
rows	`30`
wt	`json`
sort	`score desc`

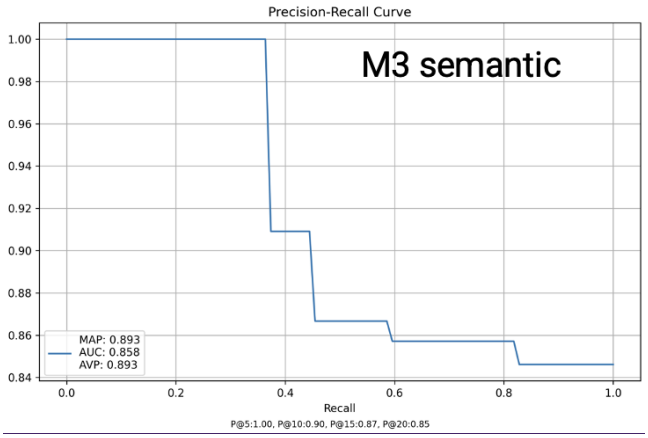

Figure 13: Visually Impressive Games semantic query

Table 31: M3 Comparison

Rank	Simple	Boosted	Semantic
1	N	N	R
2	N	R	R
3	R	N	R
4	N	R	R
5	R	R	R
6	N	R	R
7	R	R	R
8	N	R	R
9	N	R	N
10	R	R	R
P@10	0.40	0.80	0.90
AvP	0.3905	0.6636	0.9889

One more time, we get **really good results**, showing **excellent performance**, **strong overall effectiveness** and **consistent performance**. We have some drops but stepped drops, that end up stabilizing. The gradual precision drops, in our opinion, suggest that there is a good handling of subjective visual quality.

Relaxing Games

In Milestone 2, the results for relaxing games were among the lowest in all the search scenarios we chose. We are searching for games where users can **relax** after a long day, and **calm** and **peaceful** games.

Table 32: Semantic Search Query

Semantic Search Query	
q	`{!knn f=vector topK=50}relaxing peaceful calm exploration`
Description	Uses dense vectors generated by all-MiniLM-L6-v2 model to find semantically similar content. The query text is converted to a 384-dimensional vector before being sent to Solr.
fl	`id,parent_id,Title,Content,Score, paragraph_num,score`
rows	`30`
wt	`json`
sort	`score desc`

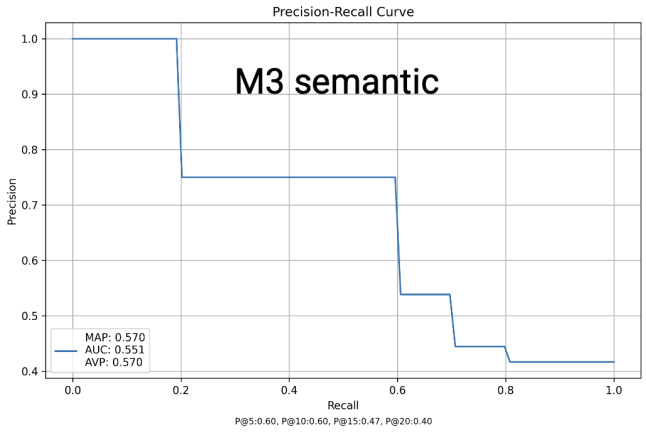

Figure 14: Relaxing Games semantic query

Table 33: M3 Comparison

Rank	Simple	Boosted	Semantic
1	R	R	N
2	N	R	R
3	R	N	N
4	N	R	N
5	N	R	R
6	N	R	N
7	N	R	R
8	N	R	N
9	R	R	N
10	R	R	N
P@10	0.40	0.90	0.30
AvP	0.6000	0.8783	0.4429

We can clearly say that the system **still struggles** when searching for relaxing games. This is without a doubt the **most challenging** search scenario for the system. The steep drops clearly indicate some difficulties in **identifying relaxing games**, and we see more volatile patterns compared to the other search scenarios. These lower results compared to the rest could be explained by the subjective nature of "relaxing". The boosted system from Milestone 3 actually got **better results** than the semantic search.

10.3 Query Rewriting

In order to improve results, we decided to try Query Rewriting. The original search terms are expanded with related terms.

Relaxing Games

For games with **tight controls**, we added "game-play, mechanics and handling", and the terms are combined using OR operators. **Title and Content** fields were given different importance so that we could prioritize where terms appear.

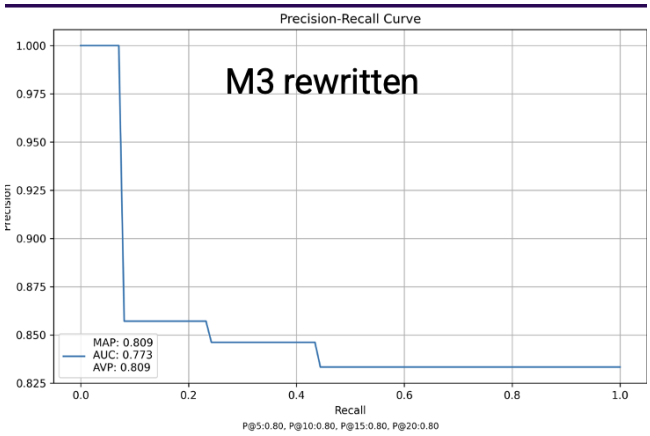

Figure 16: Games with tight controls rewritten query

We can say that both methods achieved strong results, with semantic search slightly **outperforming**. Query rewriting shows more gradual decreases in precision. Both approaches capture with success the concept of games with tight controls, maybe because the **vocabulary** is well defined and consistent, and the technical terms are **similar** across different reviews.

Multiplayer Games

For Multiplayer Games, we added "competitive, team, and co-op"

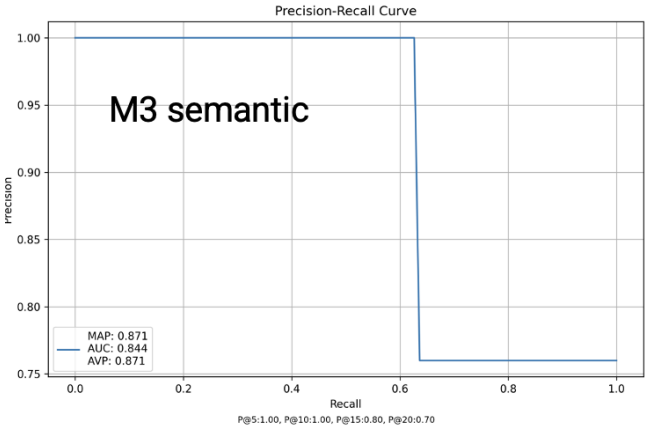

Figure 15: Games with tight controls semantic query

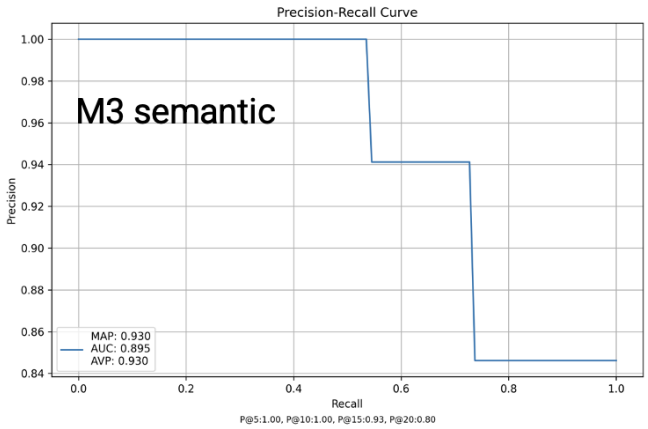

Figure 17: Multiplayer Games semantic query

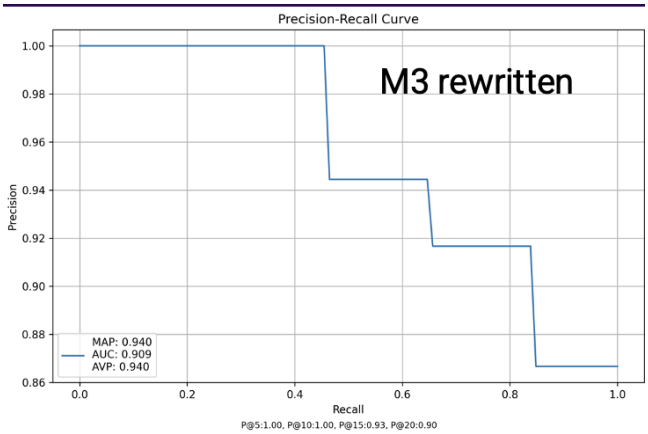

Figure 18: Multiplayer Games rewritten query

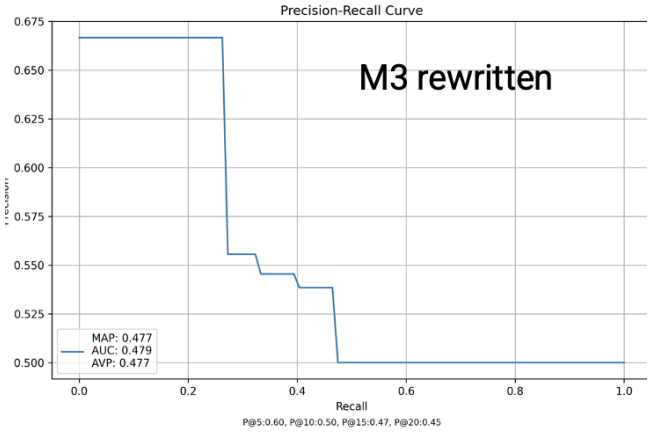

Figure 20: Story Narrative Games rewritten query

We got **really good results** for both methods. We think the similarity can be explained because the terms are **very specific** and **well defined**. The slightly better performance from the rewritten query is certainly due to the additional terms we used, capturing even more results.

Story Narrative Games

For the Story Narrative Games, we added "plot, characters, emotional".

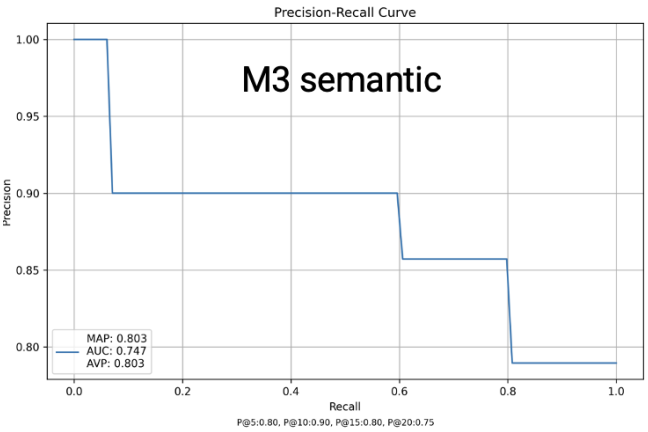

Figure 19: Story Narrative Games semantic query

This time around, there is a noticeable difference between the two approaches. The semantic search significantly outperforms query rewriting. We think this happened because narrative and story elements are **more complex and contextual**, and the semantic search understood better the relationships between the **narrative related concepts**. We can say that, at least in this case, semantic search did a better job at dealing with **complex concepts**, like in this case **Story Narrative Games**.

Visually Impressive Games

For the Visually Impressive Games, we added "resolution, fps, fidelity".

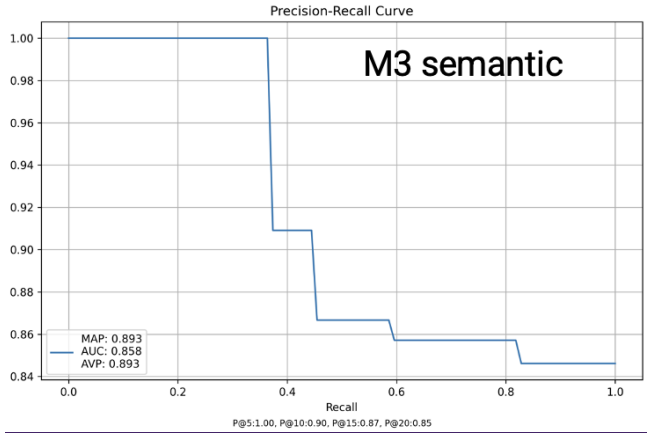

Figure 21: Visually Impressive Games semantic query

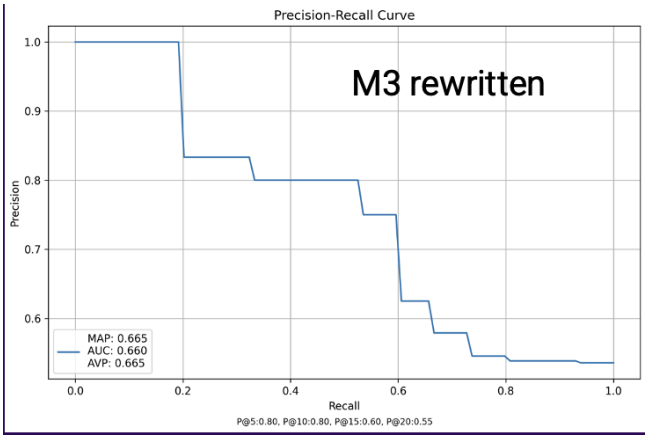

Figure 22: Visually Impressive Games rewritten query

We got again better results with the semantic search. It was **more stable** with fewer dramatic drops. This difference in performance could be explained by the fact that technical aspects of graphics can be described in various ways, and using simple terms expansions might not capture the needed results.

Relaxing Games
For the Relaxing Games, we added "casual, leisure, pastime".

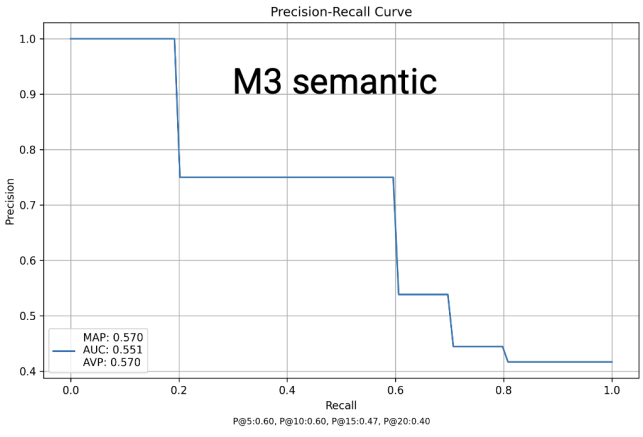

Figure 23: Relaxing Games semantic query

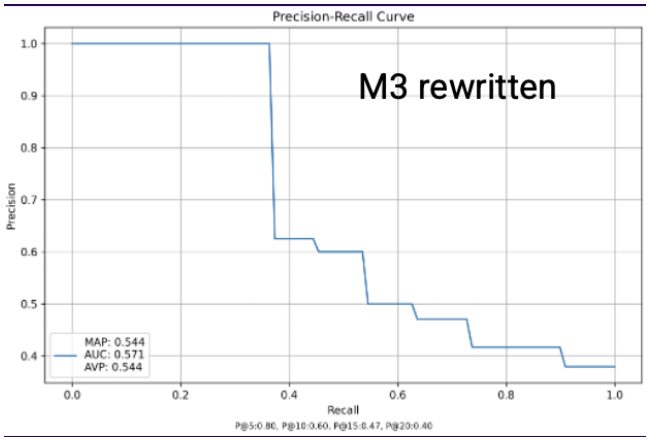

Figure 24: Relaxing Games rewritten query

Both approaches got **similar results**. There was no improvement in the Query Rewriting method. We think this is a tough concept because "relaxing" is **subjective**, as the concept of relaxation is different for everyone. A user can find a game relaxing but another user maybe doesn't find that same game relaxing. This is why performance is lower than other search scenarios.

10.4 Relevance Feedback

In order to improve the results for Relaxing Games, we decided to implement **Relevance Feedback**. Relevance Feedback works by using previously identified relevant documents, incorporating **user judgments**, and adjusting the query based on **feedback**. The documents marked as relevant are used as **positive examples** to the system. Then we expand the rewritten query with the relevant terms found in the positive examples so that we create a more refined understanding of what type of relaxing games to search.

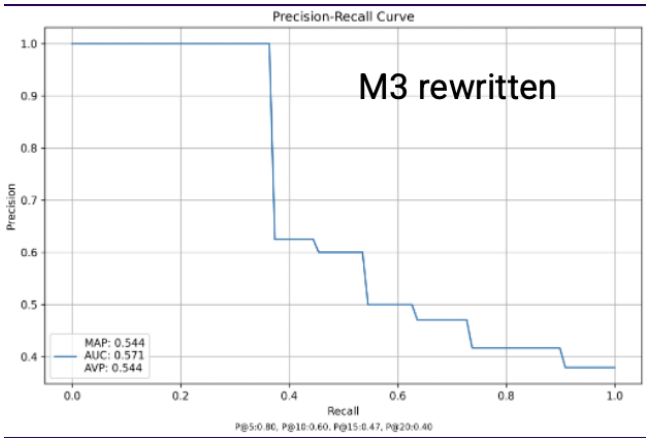

Figure 25: Relaxing Games rewritten query

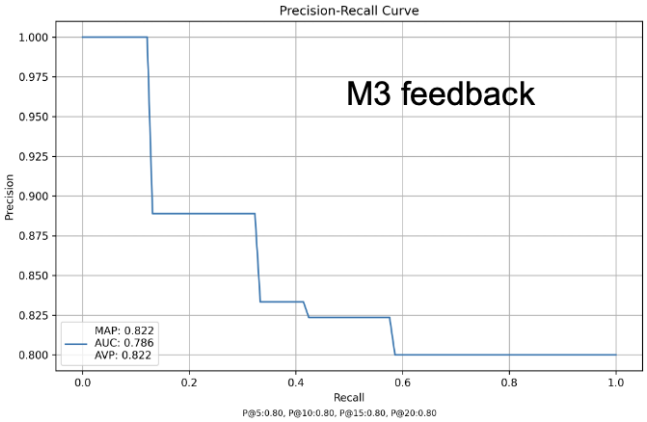

Figure 26: Relaxing Games feedback query

We can see an improvement in the results with the use of **Relevance Feedback**. It benefits from the actual user judgments on what makes a game considered relaxing. This way we can overcome the subjective nature of this search scenario.

10.5 More Like This

In order to find **similar games** after finishing reading the review of a certain game, we used the **More Like This** feature. MLT will look for games with **similar words or text**. For example, when a user is searching for a Multiplayer game, he will be recommended similar games, in the same category.

We used the document's content for similarity. We implemented **field boosting** to give different weights to different fields. and then we excluded the current review.

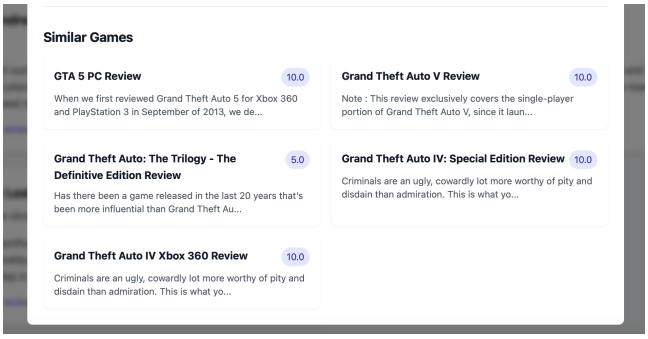

Figure 27: More Like This

We can clearly see that the system correctly presented similar games to the one we have just finished reading its review (GTA 5 PlayStation 4 and Xbox One Review).

10.6 User Interface

We decided to develop a **user interface**, alternative to the one from Solr so that we have a more intuitive way to search for game reviews. The average user certainly prefers having a **less technical** search system and **more intuitive**, that provides the same results as a **more technical** and **less intuitive** search system.

For the development of the fronted, we use several technologies, such as **Python** [14] with **Flask [12]** to run the Local Server, **JavaScript** [2] for handling the search interactions and dynamic content, **CSS** for the styling and visual presentation and **HTML** for the organization and structure of the page.

To run the project, execute the following commands in the terminal:

```
cd frontend
python3 app.py
```

This will start the Flask [12] server on `localhost:5000`, where the search interface can be accessed through a web browser.

This user interface presents several features, such as:

- **Search Filters**
  - Filter by game score
  - Filter by game category
- **Latest Reviews** – Display of most recent game reviews
- **Search Functionality** – Comprehensive search system for game reviews
- **Result Snippets and Clusters** – Preview of review content in search results and reviews grouped by clusters
- **Detailed Metadata**
  - Game score
  - Publication/update date
  - Publisher
  - Platform
  - Author
- **Similar Games** – Recommendations based on the current review

When launching the **Search System**, the user is redirected to the Home Page, where we can see the latest published reviews. We achieve this by using the **/latest** route in Flask [12]. We use the **Subheader** field to identify 2024 reviews and then randomly select 6 reviews from the results. When the page is refreshed, the game reviews are automatically fetched and updated. This provides users with direct access to recent game reviews as soon as they visit the site.

For the search part, users only need to type the game review they are looking for. They can apply **filters**, such as a **minimum score**, or **search by categories**.

After searching for the game review they want, users are presented with the **content**. **Snippets** are shown, only presenting part of the review content. The user has then the option to expand the snippets to visualize the whole review. The snippets provide enough context for the user to decide if he wants to expand the review to its full size. After expanding the review, users get access to **more information**, such as the **score**, **publish/update date**, **publisher**, **author** and **platform** (when available). After reading the whole review, the user is presented with **similar games**, so that he can carry on reading reviews about related games, reviews that the user might also enjoy.

This is a user interface that has a **responsive design** for different screen sizes, has **error handling** and **user feedback**, for example when no reviews are found. It is a **clean** and **intuitive interface**, that makes it easy for the user to search for game reviews.

10.7 Conclusion

We now reach the end of the latest Milestone 3. We successfully implemented **semantic search**, while also **improving** the search accuracy from the previous Milestone. We created a **modern** and **intuitive user interface**, with **additional features**, such as **filters**, **snippets** and **similar games**. We compared the **simple** and **boosted** setups we had, as well as comparing the impact of **semantic search** on our results, which in general, was quite positive, even if not for all search scenarios cases. We now have a **visual presentation** of the results, allowing users to **quickly** and **easily** access the reviews from the games they like. We could improve even more and implement more features, but we think we already have created a **stable** and **very usable** search system.

References

[1] Douglas Crockford. 2001. JSON: JavaScript Object Notation. https://www.json.org/. Accessed: 2024-11-18.
[2] Brendan Eich. 1995. JavaScript: A high-level, interpreted programming language. https://developer.mozilla.org/en-US/docs/Web/JavaScript. Accessed: 2024-12-17.
[3] Elasticsearch. 2024. EDisMax Query Parser: Extended DisMax for Advanced Querying. https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-dis-max-query.html. Accessed: 2024-11-18.
[4] John D. Hunter et al. 2007. Matplotlib: A 2D Graphics Environment. https://matplotlib.org/. Accessed: 2024-10-15.
[5] Michael Waskom et al. 2012. Seaborn: Statistical Data Visualization. https://seaborn.pydata.org/. Accessed: 2024-10-15.
[6] Apache Software Foundation. 2006. Apache Solr: Open Source Enterprise Search Platform. https://solr.apache.org/. Accessed: 2024-11-18.
[7] Apache Software Foundation. 2024. Solr Filters: Token Filters, Query Filters, and More. https://solr.apache.org/guide/. Accessed: 2024-11-18.
[8] Jason Huggins. 2004. Selenium. https://www.selenium.dev
[9] IGN Entertainment, Inc. 1996. IGN: Video Game Reviews, News, and Trailers. https://www.ign.com/. Accessed: 2024-10-15.
[10] Kenneth Reitz. 2011. Requests Library. https://pypi.org/project/requests/
[11] Leonard Richardson. 2004. BeautifulSoup. https://pypi.org/project/beautifulsoup4/
[12] Armin Ronacher. 2010. Flask: A lightweight WSGI web application framework. https://flask.palletsprojects.com/. Accessed: 2024-12-17.
[13] The Pandas Development Team. 2008. Pandas: Python Data Analysis Library. https://pandas.pydata.org/. Accessed: 2024-10-15.
[14] Guido van Rossum. 1991. Python. https://www.python.org

A Annexes

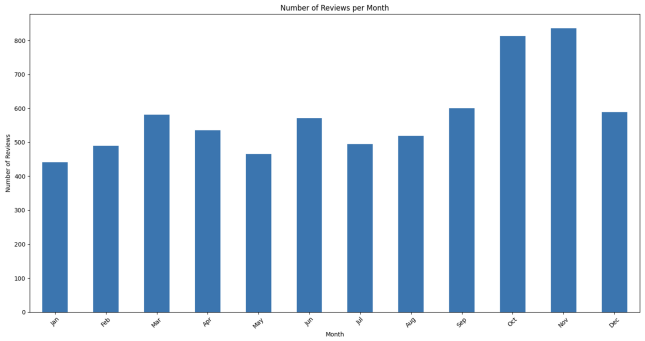

Figure 28: Reviews per Month

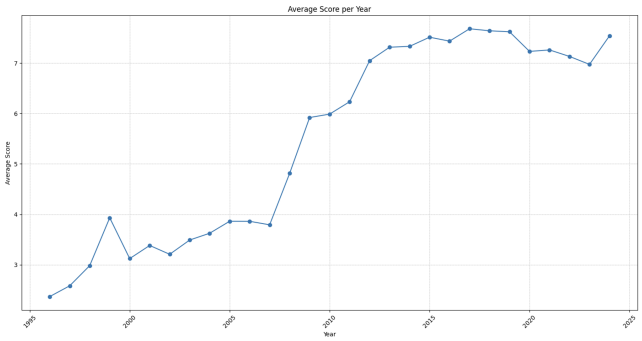

Figure 29: Scores per Year

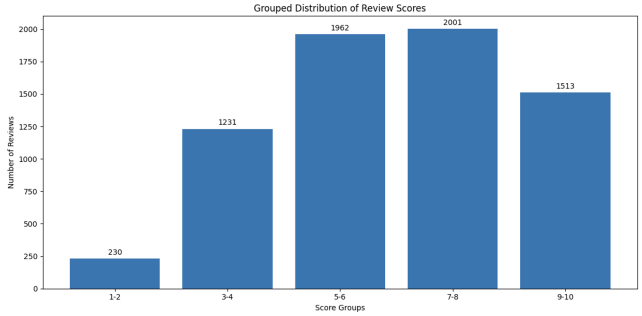

Figure 30: Distribution Review Scores

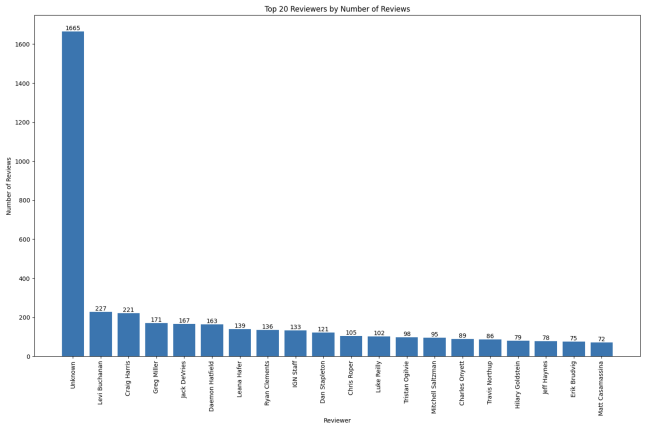

Figure 31: Top20 Reviewers

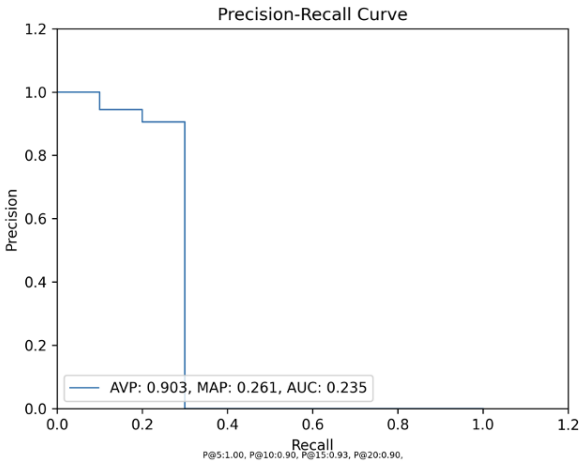

Figure 35: Games with Tight Controls (simple)

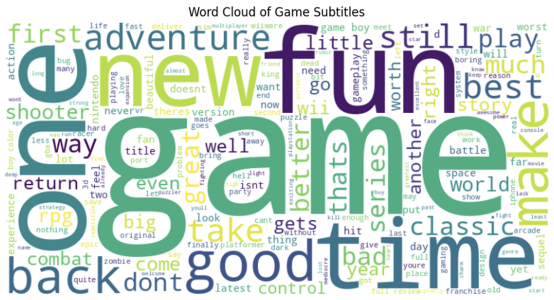

Figure 32: Word Cloud for Subtitles

Figure 33: Word Cloud for Content

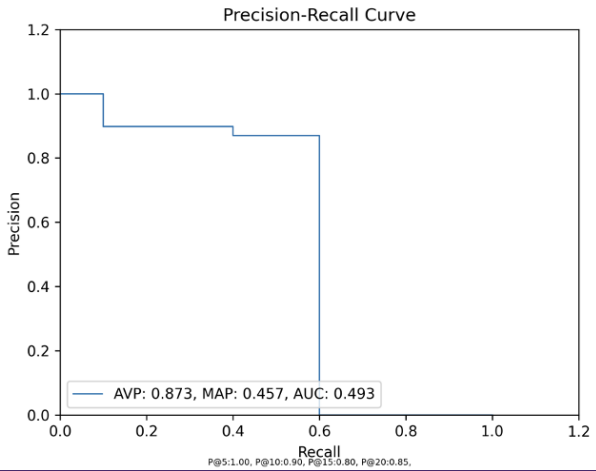

Figure 36: Games with Tight Controls (boosted)

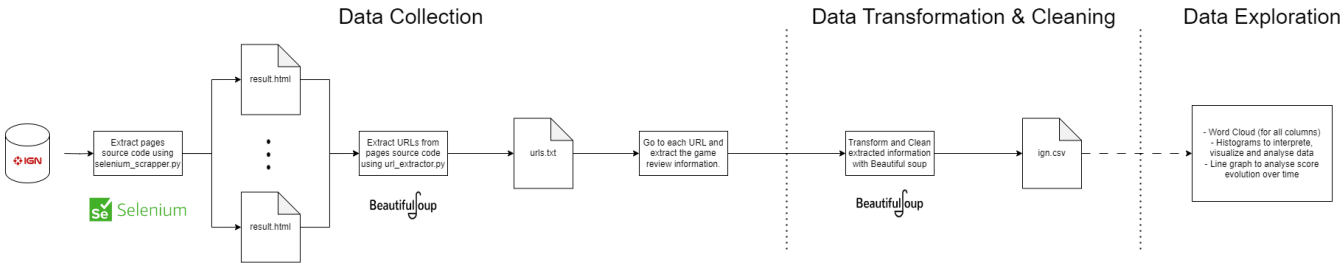

Figure 34: Data Pipeline

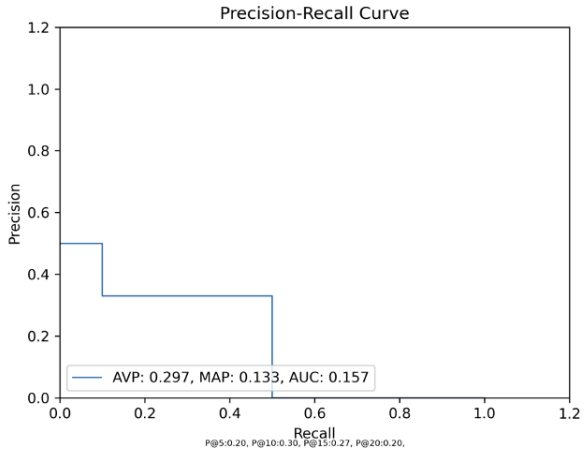

Figure 37: Games good for Relaxing (simple)

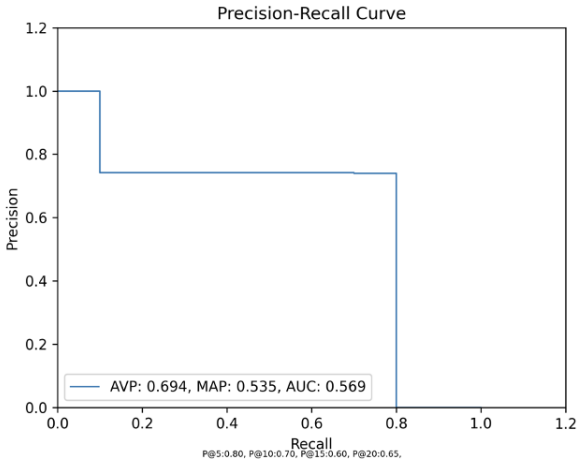

Figure 39: Story Narrative Games (simple)

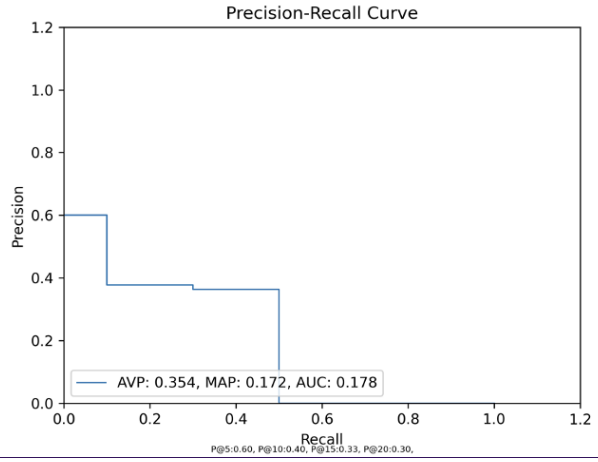

Figure 38: Games good for Relaxing (boosted)

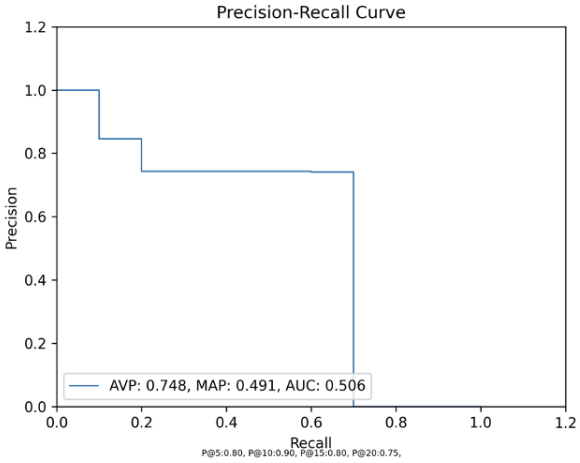

Figure 40: Story Narrative Games (boosted)

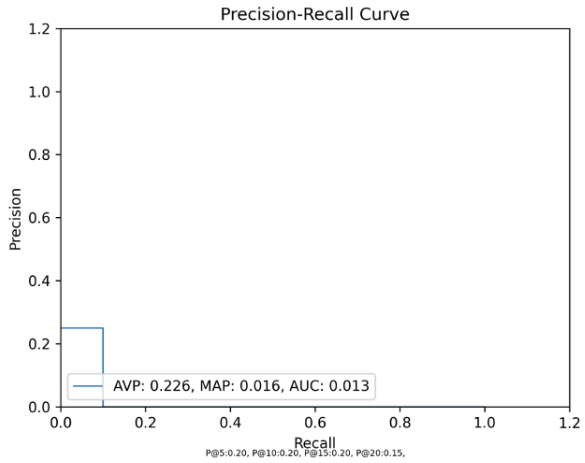

Figure 41: Visually Impressive Games (simple)

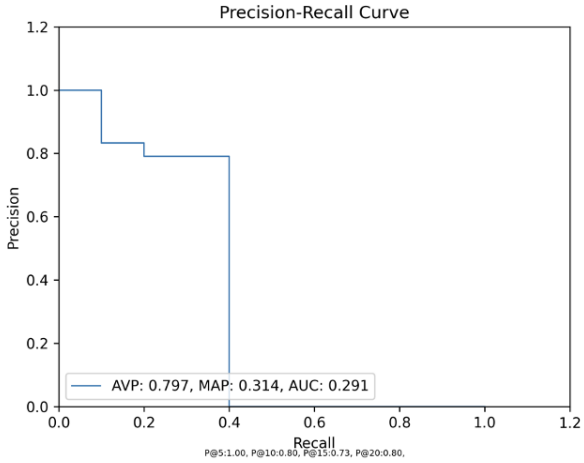

Figure 43: Multiplayer Games (simple)

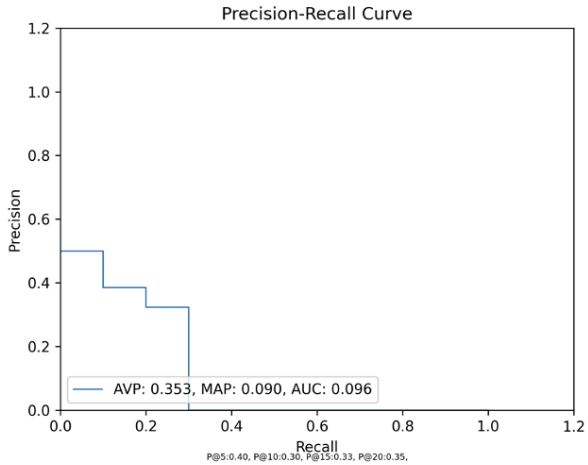

Figure 42: Visually Impressive Games (boosted)

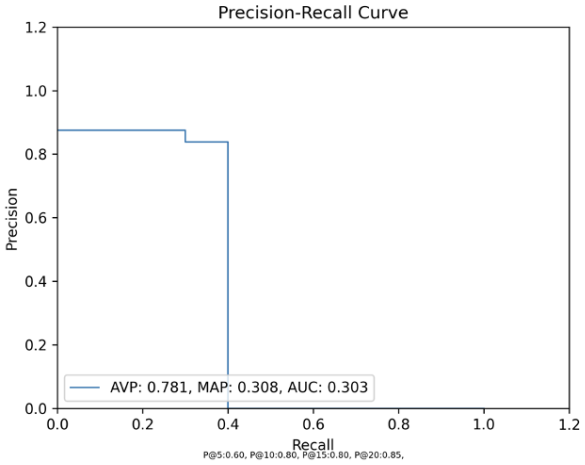

Figure 44: Multiplayer Games (boosted)

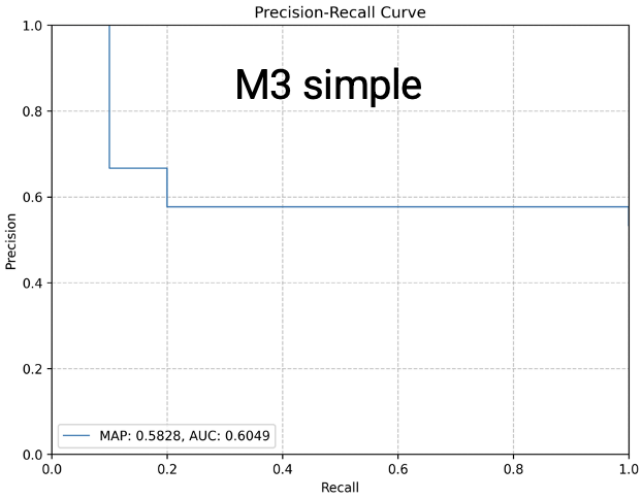

Figure 45: Games with tight controls (simple)

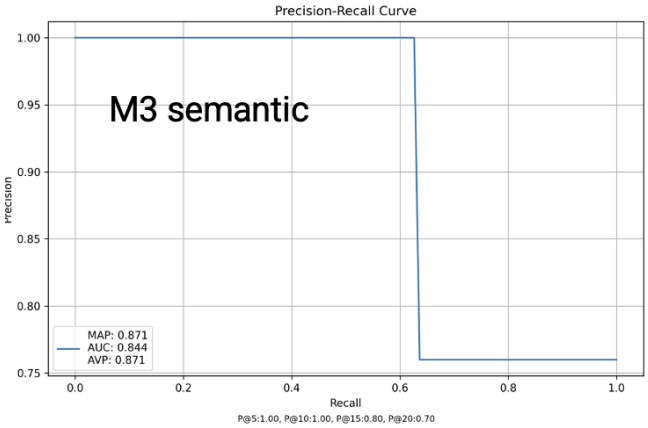

Figure 47: Games with tight controls (semantic)

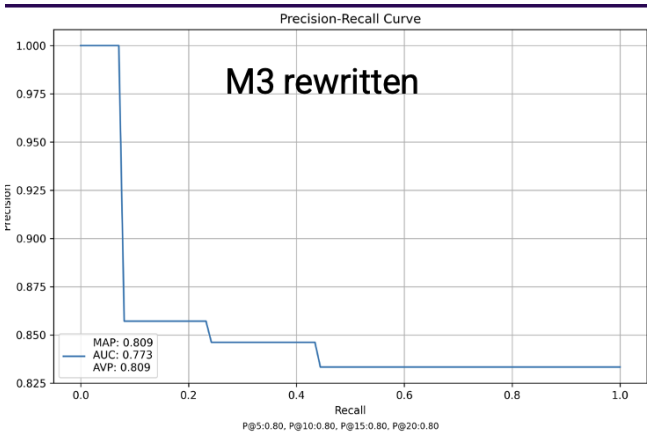

Figure 48: Games with tight controls (rewritten)

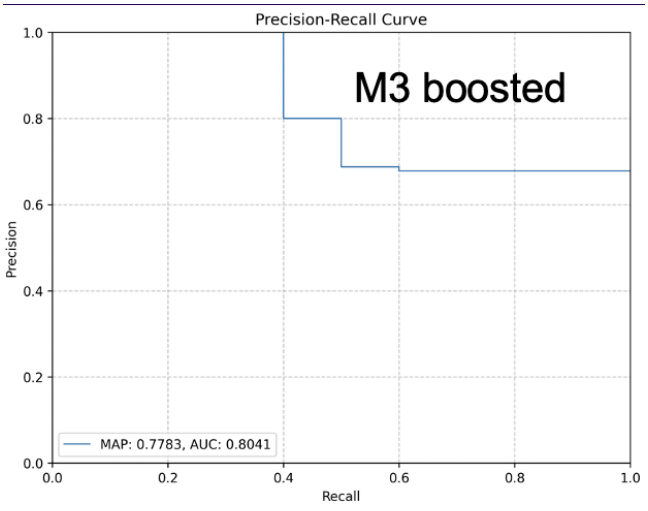

Figure 46: Games with tight controls (boosted)

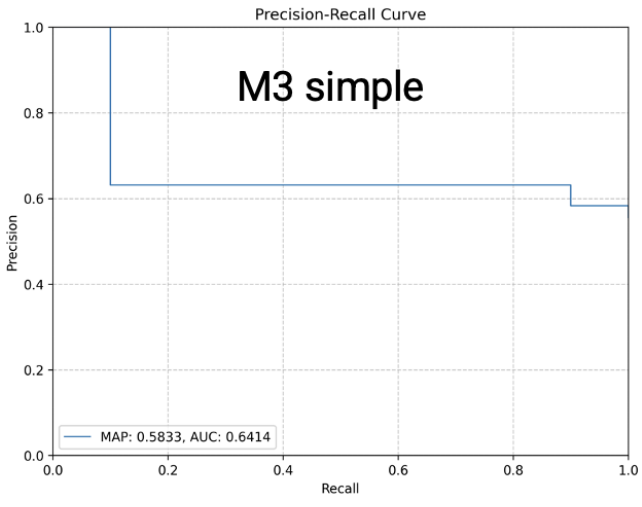

Figure 49: Multiplayer Games (simple)

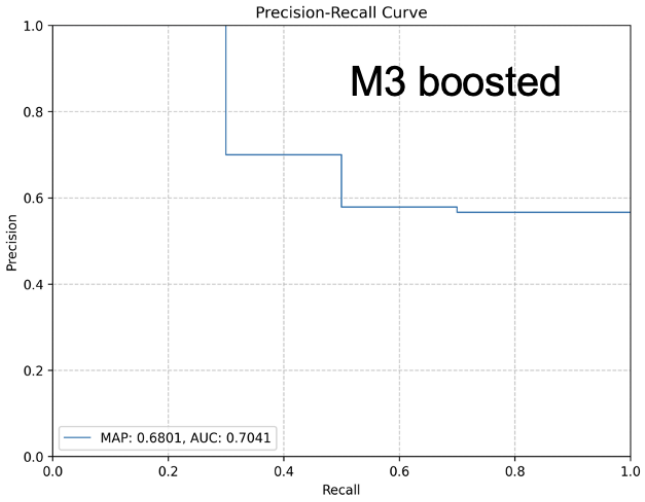

Figure 50: Multiplayer Games (boosted)

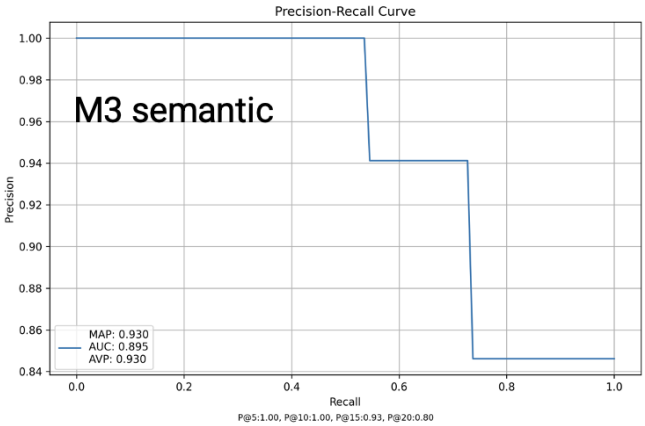

Figure 51: Multiplayer Games (semantic)

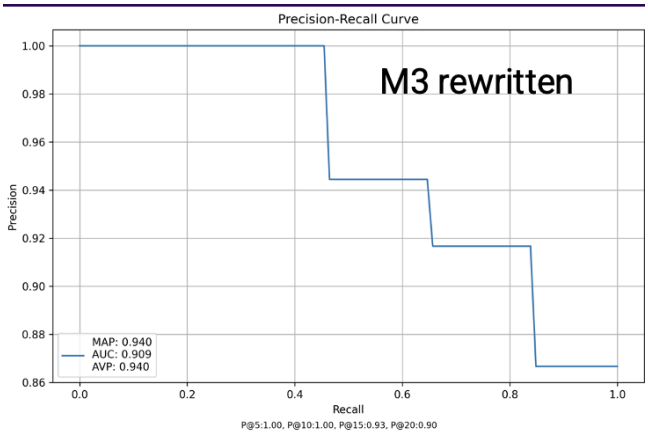

Figure 52: Multiplayer Games (rewritten)

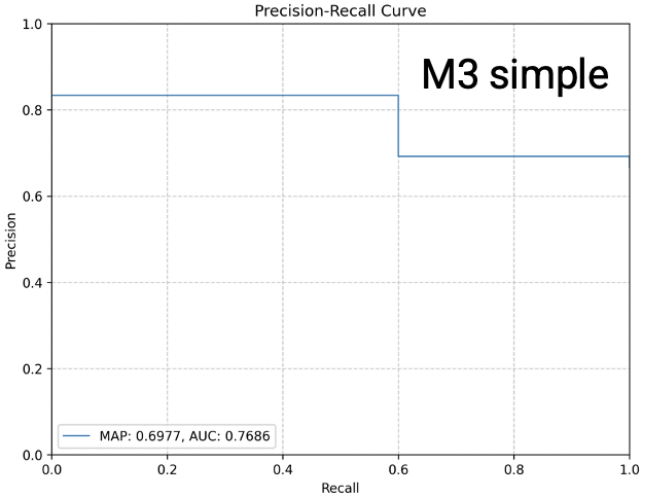

Figure 53: Story Narrative Games (simple)

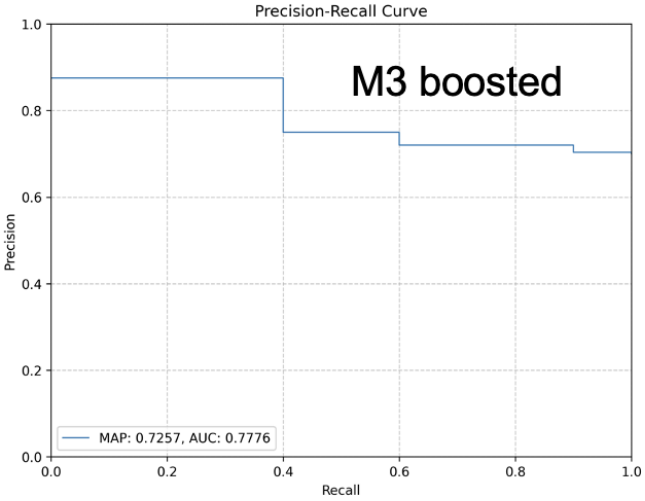

Figure 54: Story Narrative Games (boosted)

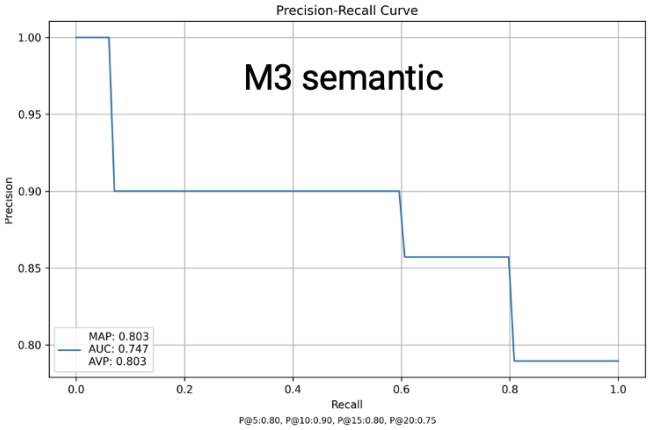

Figure 55: Story Narrative Games (semantic)

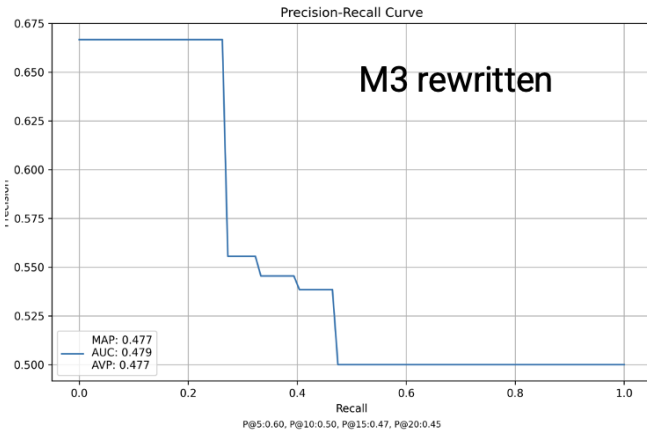

Figure 56: Story Narrative Games (rewritten)

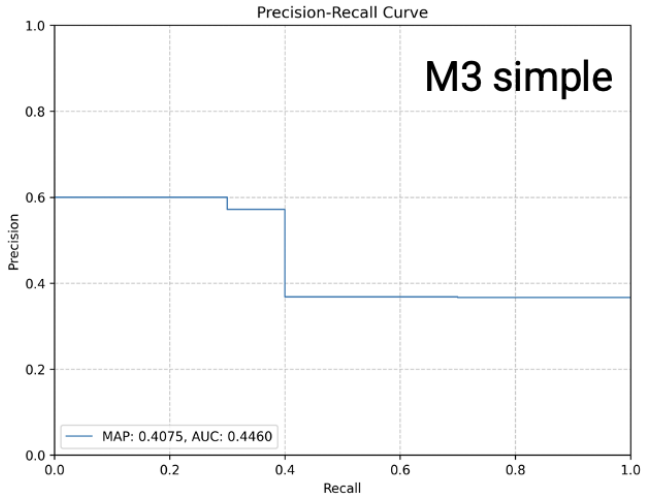

Figure 57: Visually Impressive Games (simple)

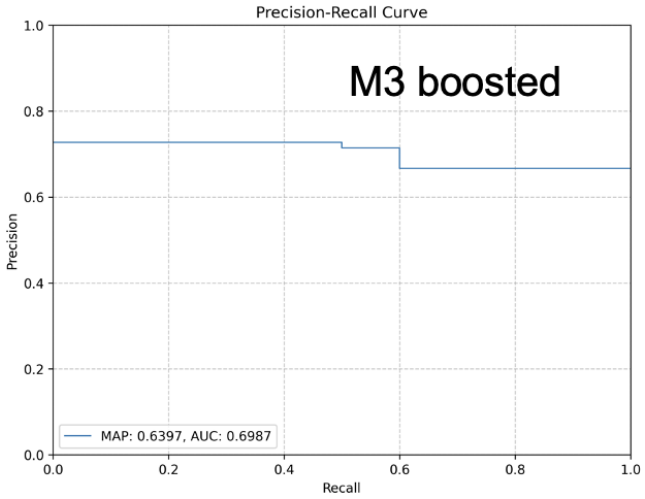

Figure 58: Visually Impressive Games (boosted)

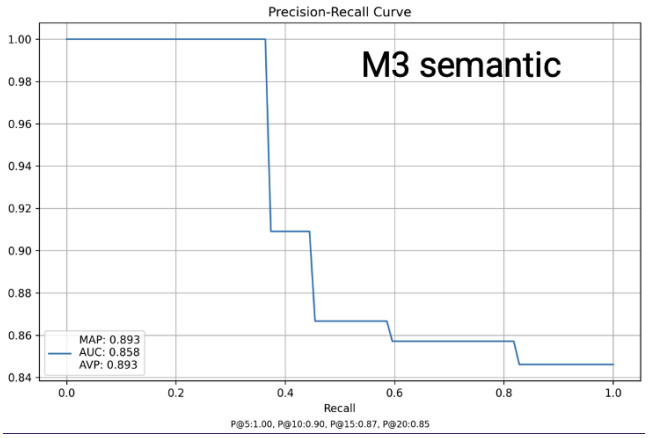

Figure 59: Visually Impressive Games (semantic)

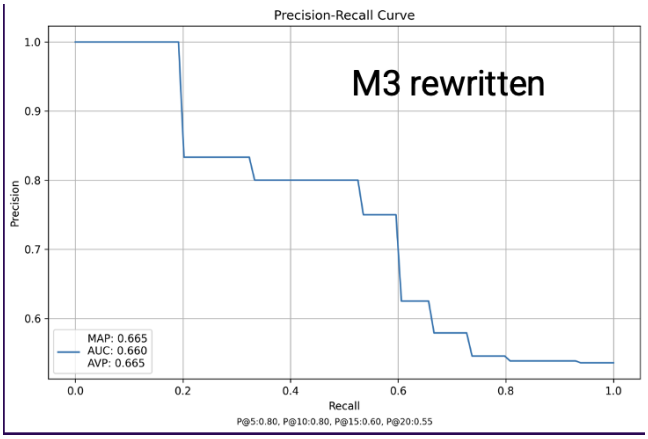

Figure 60: Visually Impressive Games (rewritten)

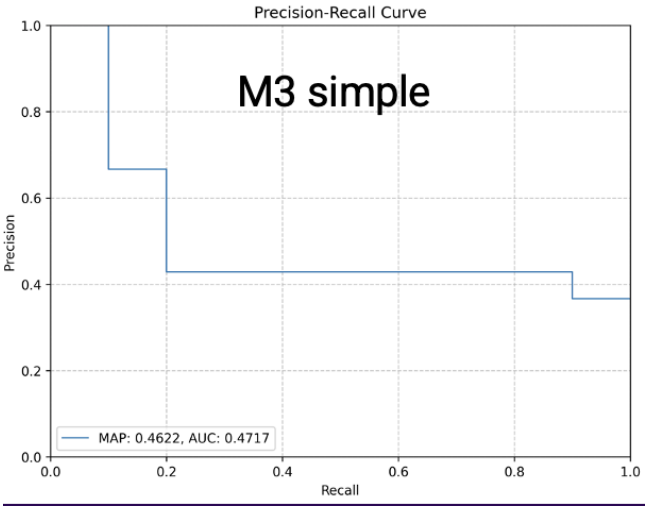

Figure 61: Relaxing Games (simple)

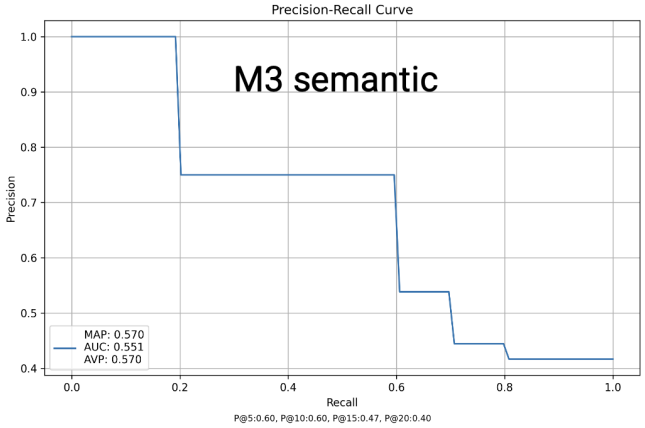

Figure 63: Relaxing Games (semantic)

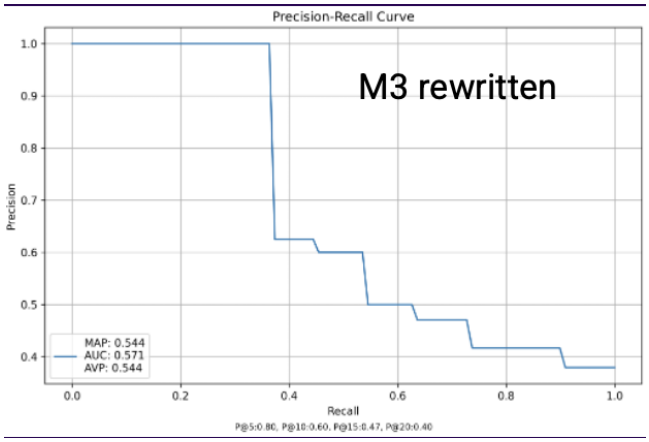

Figure 64: Relaxing Games (rewritten)

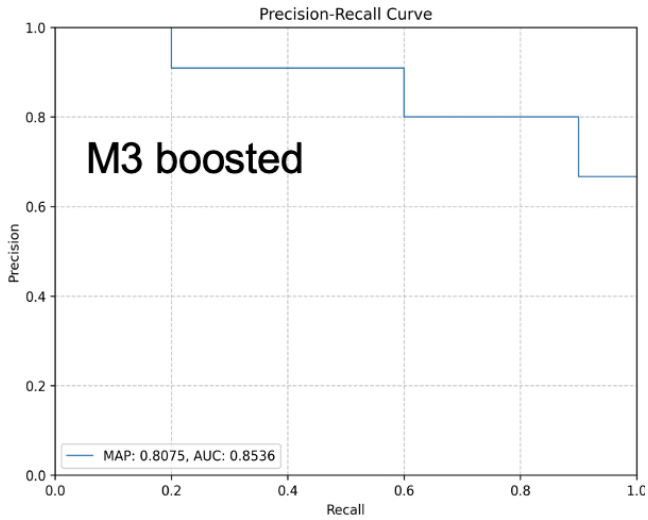

Figure 62: Relaxing Games (boosted)

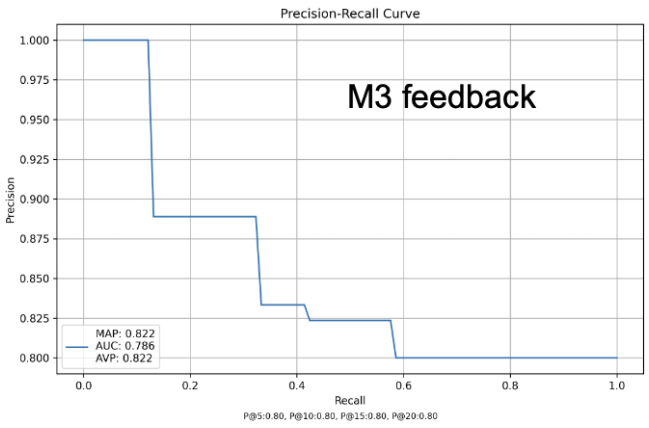

Figure 65: Relaxing Games (feedback)