685 Mental Disorders: Search System

Francisco Ribeiro
up202104797@fc.up.pt

Marisa Azevedo
up202108624@fc.up.pt

Toni Grgurevic
up202408625@fe.up.pt

Abstract

This report focuses on the creation of an information retrieval system for a document collection containing 685 mental disorders extracted from Wikipedia. Using techniques like web scraping and API data extraction, we assembled a structured dataset with detailed attributes, including symptoms, treatments, and metadata such as revision history and page views. To achieve this, a pipeline was built with five main phases: data collection, data cleaning, data enrichment, data structuring, and data exploration. As a result of this process, a document collection of approximately 18.5 MB was generated. The next step was indexing the dataset using two schemas: a very simple schema and a more complex schema. The complex schema incorporated advanced filters, such as synonym expansion. The evaluation revealed that the simple schema achieved a Mean Average Precision (MAP) of 0.620, while the complex schema demonstrated improved performance with a MAP of 0.714. This report also outlines the advancements planned for Milestone 3, focusing on semantic search capabilities, the integration of Rocchio algorithms for query refinement, and the evaluation of performance improvements for these newly integrated features. Finally, we also built a user interface.

The full implementation details and source code can be accessed at [5].

CCS Concepts: • **Information Systems** → Information Retrieval.

Keywords: Web Scraping, API Data Extraction, Information Retrieval, Pipeline Documentation

1 Introduction

Understanding mental health conditions is essential for advancing research, increasing awareness, and offering better support to those affected. With the information available online, particularly on platforms like Wikipedia, there is a chance to organize and structure this data in a more accessible format. In this project, we aim to collect 685 mental health disorders from Wikipedia, using techniques like API data extraction and web scraping to create a structured and comprehensive document. This document collection will serve as a foundation for building an information retrieval system designed to aid mental health research and analysis.

Information Processing and Retrieval, FEUP, 2024

The document is organized as follows: Section 2 describes the data sources used for the collection. Section 3 explains the five key phases of the data preparation pipeline, including data collection, cleaning, enrichment, structuring, and exploration. Section 4 characterizes the document collection, discussing its key features, structure, and potential prospective search tasks. Section 5 outlines the information retrieval system, covering the schemas, indexing, and retrieval processes. Section 6 evaluates the system's performance, including an analysis of the results obtained from sample queries designed to test the retrieval capabilities. Section 7 concludes the milestone 2 report by summarizing the key findings and providing insights for future work. Section 8 describes the integration of semantic search and Rocchio algorithms for enhanced retrieval performance. Section 9 evaluates the improvements achieved in this milestone. Section 10 introduces the user interface designed to facilitate interaction with the system. Finally, Section 11 provides a comprehensive characterization of the final system, and Section 12 outlines the conclusions and potential directions for future work. Finally, the appendix provides additional figures and visualizations.

2 Data Sources - Wikipedia

In this project, the primary source of information for mental [9] and neurological disorders [10] was Wikipedia. As one of the largest, most up-to-date, and freely accessible repositories of knowledge on the web, Wikipedia offers extensive information on a wide variety of medical and psychological topics, including mental health disorders. By utilizing both API data extraction and web scraping techniques, we were able to gather comprehensive data from Wikipedia.

As Wikipedia operates under the "Creative Commons Attribution ShareAlike 4.0 International License (CC BY-SA 4.0)" [8], this project ensured full compliance with its licensing requirements. This license allows for the sharing and adaptation of content for any purpose, including commercial use, provided proper attribution is given and any derivative works are distributed under the same license.

2.1 Reliability and Data Quality

Articles on Wikipedia are authored and edited collaboratively, which can result in varying levels of accuracy, completeness, and consistency. For this project, these factors were carefully considered to ensure the credibility of the collected dataset.

- **Varying Article Completeness:** Articles on mental and neurological disorders differ significantly in detail.

Some entries are well-researched and cited, while others lack depth or include incomplete sections. In particular, certain disorders, especially rare or less-studied ones, have minimal information available, often missing critical attributes like *causes* or *treatments*.

- **Metadata Analysis:** Fields such as revision history, number of edits, and page views were collected to assess the stability and popularity of articles. Articles with higher edit counts and more frequent updates were prioritized for closer inspection.

2.2 API Data Extraction

Wikipedia provides a robust API (Application Programming Interface), which facilitates automated access to its content.

For this project, we used the API to extract basic information for each disorder, such as, a summary (description), the Wikidata ID and the number of revisions, which can provide insights into the history and credibility of the article.

2.3 Web Scraping with BeautifulSoup

While the API was helpful for extracting structured data, not all the necessary information was readily available via this method. To overcome these limitations, we employed web scraping techniques using the Python library BeautifulSoup [4]. Web scraping involves programmatically accessing a webpage's HTML structure and extracting the desired content.

Through BeautifulSoup, we gathered detailed information from each Wikipedia mental disorder page that was not accessible via the API, such as sections: the causes, symptoms, treatments, and epidemiology of each disorder.

3 Data Collection and Preparation (Milestone 1)

The process of collecting, preparing, and structuring the data for this project was organized into several key phases, which are illustrated in 9 in the Appendix. The pipeline begins with the data collection phase, where information is gathered from Wikipedia using both API data extraction and web scraping techniques. This is followed by data cleaning, where duplicate and incomplete entries are removed, and data enrichment, which includes adding links to Wikidata and further information from additional sources.

3.1 Data Collection

The first phase of the pipeline involves gathering raw data on 685 mental and neurological disorders from Wikipedia pages [9] and [10]. Both types of disorders were stored in separate JSON files for better organization.

3.2 Data Cleaning

After the initial data collection, the next phase focuses on cleaning the collected data to ensure quality and consistency. During this step, we:

- Remove duplicates: Disorders that appeared on both Wikipedia pages were consolidated into a single entry to ensure only one complete version was retained for each disorder.
- Correct missing data: We noticed some information about certain disorders was incomplete, so we found the bug that was causing the problem and handled that.
- Content Formatting: The raw text contained citation markers (e.g., "[1]", "[2]"), which we removed to improve readability and ensure cleaner, more structured data.

3.3 Data Structuring

We decided to structure the data in a way that captures all relevant information about each disorder while ensuring consistency across entries. The structured data schema for each disorder is illustrated in 1.

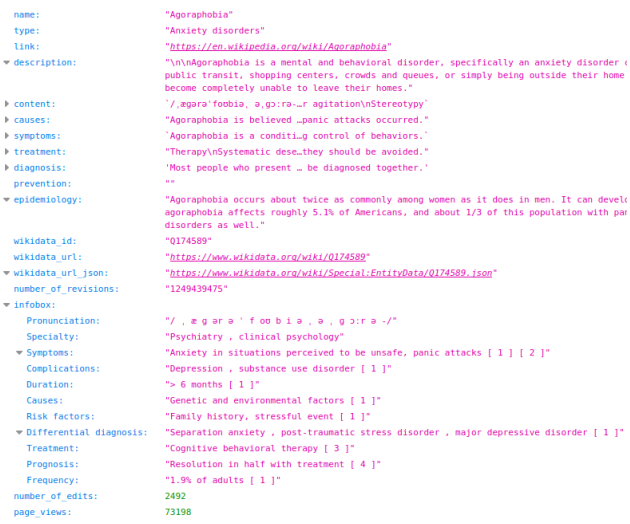


Figure 1. Data Structure

Below are the key components of this structure:

- **Name**: The official name of the disorder as it appears on Wikipedia.
- **Type**: The classification or category of the disorder (e.g., "Anxiety disorders").
- **Link**: The URL link to the full Wikipedia article for further details.
- **Description**: A brief, summary of the disorder, providing essential information.
- **Content**: The full textual content extracted from the article, offers a more in-depth explanation and context.

- **Causes**: A description of the factors or events believed to contribute to the disorder.
- **Symptoms**: The main symptoms or characteristics that are typically observed in individuals with the disorder.
- **Treatment**: Information about available treatments, therapies, or interventions for managing the disorder.
- **Diagnosis**: Criteria and methods used to diagnose the disorder.
- **Prevention**: Any noted strategies or practices aimed at preventing the disorder.
- **Epidemiology**: Statistical data and prevalence information, highlighting how common the disorder is and any relevant demographic factors.
- **Wikidata ID, URL, JSON**: These fields link the disorder to its corresponding Wikidata entry.
- **Number of Revisions**: This field tracks the number of revisions made to the Wikipedia article, giving an indication of its revision history.
- **Infobox**: The infobox is a structured set of additional attributes that offer a quick reference for key aspects of the disorder.
- **Number of Edits**: This field captures how many times the Wikipedia article has been edited, which can indicate the level of activity.
- **Page Views**: The total number of views the Wikipedia article has received in the last 30 days, providing insight into its popularity or relevance over time.

3.4 Corrections

During the data processing phase, we noticed several disorders with missing information, some were because they did not exist, others because the script was not detecting the sections, after some hours trying to understand the situation, we found out that some class names were very common and it pointed to other content that we did not want so we had to specify a little more while doing the scraping.

3.5 Final Data

After passing through the phases of collection, cleaning, structuring, and enrichment, the final dataset was stored in JSON format. This structured format allows for clear visualization and easy navigation of the data, making it ideal for further exploration and analysis. The dataset includes not only the essential attributes such as disorder name, type, and description, but also enriched metadata like the number of revisions, detailed sections on symptoms and treatments, and links to external resources.

4 Data Characterization

In this section, we describe the document used to build the search engine, consisting of 685 documents related to different disorders. Each document contains multiple attributes, and various optional fields. The primary goal of this analysis is to explore the structure and distribution of the data in terms of document length, attribute completion (distribution of empty fields), and other characteristics.

4.1 Domain Model

To better understand the relationships between the key components of the dataset, we created a conceptual data model, as shown in Figure 2.

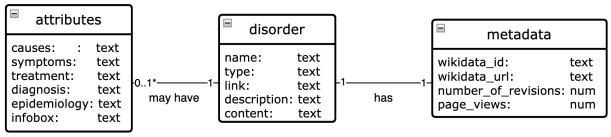


Figure 2. Conceptual Data Model

The model includes the following key entities:

- **Disorder:** The central entity, representing each mental or neurological disorder with attributes like `name`, `type`, `link`, and `content`.
- **Attributes:** Optional fields such as `causes`, `symptoms`, `treatment`, and `epidemiology`, which may or may not be present for all disorders.
- **Metadata:** Information about the reliability and popularity of articles, including `wikidata_id`, `number_of_revisions`, and `page_views`.

The relationship between **disorder** and **attributes** is optional (0..1), as not all disorders have detailed attributes. Meanwhile, the relationship between **disorder** and **metadata** is mandatory (1:1), as every disorder includes metadata.

4.2 Document length analysis

One key aspect of the dataset is the length of documents for each disorder. Figure 10 shows the histogram of document lengths. The distribution is highly skewed, with many documents being relatively short, but there are also a few outliers with very high word counts. The figure also shows the mean and median value.

To further explore the variability in document length, Figure 11 presents a box plot. The plot highlights a large variance between documents, with a significant number of outliers having much longer text than the majority. This variance could be due to the nature of the disorders or the completeness of the information provided.

We also examined the differences in document length between different disorder types. Figure 12 shows the mean word count for each type. The analysis reveals that some disorder types tend to have longer or more detailed documents, which may reflect the complexity of the condition or the availability of information.

Most of the documents (disorders) are of type "neurological disorders", 385 different disorders of this type. This distribution may influence the overall performance of the search engine if certain types of disorders are overrepresented.

Figure 13 shows a pie chart of how the document length is distributed across attributes. This helps to understand where most of the content lies. Attributes that are short (one sentence) or not of text type are left out for a clearer chart. Content is clearly the most populated attribute since information that can't be placed in other optional fields ends up there.

4.3 Null Values

An important factor in understanding the completeness of the dataset is the presence of null (empty) fields. Figure 14 presents the distribution of null values across the different attributes. There are 2016 empty fields in total, with certain attributes being more commonly left out than others since not all disorders had sections for that attribute. The most common empty attribute is prevention which could be even left out of the document.

Since it could also be interesting to see how many empty fields are in documents based on types, we present that in Figure 15. Most of the empty fields are in neurological-type disorders which is understandable since most of the disorders in the data are of that type. The unexpected result is a relatively high count of null values in documents with type substance-related disorders

4.4 Word Cloud

To capture the common themes and terms present across the dataset, we generated a word cloud. Figure 16 visualizes the most frequent words across all documents, emphasizing the key concepts and terminology used in describing various disorders.

4.5 Summary

The data characterization revealed that the dataset is highly diverse in terms of document length and attribute completion. Certain types of disorders, such as neurological disorders, are overrepresented, and optional attributes like prevention, epidemiology and causes are often missing. This variability may impact the search engine's performance, and future iterations could focus on improving data completeness and balance across disorder types.

4.6 Prospective Search Tasks

After gathering and organizing the document collection, it is crucial to determine the types of queries the system should address. The search system will assist users in exploring the dataset and retrieving relevant information about mental disorders. Some relevant search scenarios are as follows:

- Find disorders where cognitive speed is significantly affected.
- Search for disorders commonly associated with childhood trauma.
- Identify disorders that respond well to behavioral therapies.
- Retrieve disorders frequently diagnosed in children.
- Explore disorders caused by genetic inheritance.

5 Information Retrieval (Milestone 2)

Information Retrieval (IR) is the process of searching and retrieving relevant information from large collections of data.

The development of an effective IR system involves several key steps, including defining documents, indexing data, selecting relevant fields, and creating schemas to organize information.

For this project, Apache Solr [2] has been chosen as the primary tool due to its powerful features for full-text search, scalability, and flexibility in handling complex data, and since it was not the purpose of this project to explore other tools.

This section outlines the process of building the retrieval system using Apache Solr, covering document definition, indexing, and schema creation.

5.1 Document Definition

Each document of the search system represents a mental health disorder and contains structured and unstructured data attributes resulting from the process of data extraction and all the steps mentioned above.

5.2 Indexing Process

Indexing is a crucial step in the Information Retrieval (IR) pipeline, as it organizes the data to optimize search efficiency. Without indexing, searching through large datasets would be slow and computationally expensive. In Solr, indexing is achieved through Tokenizers and Filters. Tokenizers break the text into smaller units, or tokens, which can be processed, while Filters modify and standardize these tokens for more efficient searching.

The default indexing provided by Solr was not able to correctly index the data, for that reason we developed a simple schema, with just some small corrections in types, to have a base point for improvement, and then we developed a custom and more complex schema, which is the main focus of this section.

The idea regarding improvement was to index textual fields, such as disorder descriptions, symptoms, and treatment information, as they carry the most context and information relevant for searches. Other fields, such as metadata or unique identifiers, are stored but are not indexed, as they are not the focus of the search process in this particular system, and, in this way, it also permits a reduction in indexing overhead.

A custom indexing analyzer was developed to handle the textual fields. This analyzer incorporates various stages, which are detailed below for both the 'custom_text_general' and 'text_phonetic' field types:

- **StandardTokenizerFactory**: This tokenizer splits the text based on spaces and punctuation, ensuring that individual words and terms are properly isolated for indexing.
- **ASCIIFoldingFilterFactory**: This filter normalizes characters, converting accented characters into their equivalent ASCII form (e.g., "é" becomes "e"), ensuring consistency in searches.
- **LowerCaseFilterFactory**: This filter converts all characters to lowercase, making searches case-insensitive.
- **SynonymGraphFilterFactory**: A custom synonym filter was applied to expand each token to include its synonyms, ensuring that queries match different terms that convey the same meaning in the context of diseases (e.g., "depression" and "sadness" or "child" and "youngster"). To construct a comprehensive dictionary of synonyms related to various diseases, we utilized OpenAI's ChatGPT. The prompt used was as follows: "I want a dictionary of synonym terms related to the following themes: Symptoms and Emotions, Situational Triggers, Psychological Aspects, Treatment Methods, Co-occurring Conditions, Cultural and Social Factors, and Theoretical Frameworks. The context for extracting words includes all disorder 01 data, all disorder 02 data, all disorder 03 data." This approach allowed us to generate a robust set of synonyms pertinent to the specified themes and disorders.
- **EnglishMinimalStemFilterFactory**: This filter reduces each token to its root form (stemming), so variations of a word (e.g., "treatments" and "treat") are treated as the same term.

These steps are applied in the 'custom_text_general' field type, which is primarily used for textual data that does not involve phonetic variations (e.g., descriptions, causes, and symptoms).

However, for fields that require handling of phonetic variations (e.g., disorder names), the 'text_phonetic' field type is employed. This field type uses a different set of filters, including the phonetic filter, which is particularly useful for fields like disorder names that may be misspelled or have alternative spellings based on pronunciation.

- **PhoneticFilterFactory** with *encoder= "DoubleMetaphone"*: This filter generates a phonetic representation of the term using the Double Metaphone algorithm, which helps in matching terms with similar sounds but different spellings (e.g., "schizophrenia" vs. "scizophrenia").
- **Other Filters**: Similar to the 'custom_text_general' field type, the 'text_phonetic' type also uses the *LowerCaseFilterFactory* and `StandardTokenizerFactory` to ensure consistency in indexing and querying.

The indexing process for the mental health disorder documents can be summarized in the table 1.

Table 1. Schema Field Types

Field	Type	Indexed
name	text_phonetic	yes
type	string	yes
link	string	no
description	custom_text_general	yes
content	custom_text_general	yes
causes	custom_text_general	yes
symptoms	custom_text_general	yes
treatment	custom_text_general	yes
diagnosis	custom_text_general	yes
epidemiology	custom_text_general	yes
wikidata_id	string	no
wikidata_url	string	no
number_of_revisions	pint	yes
page_views	pint	yes
infobox	custom_text_general	yes

5.3 Search System Configuration and Retrieval Process

To evaluate the system's ability to retrieve relevant documents, we set up a retrieval process using the Solr search engine. The system was configured to handle queries related to mental health disorders by leveraging the two, already mentioned, distinct schemas: the **simple schema** and the **complex schema**.

The retrieval process was fine-tuned using the **edismax** [1] query parser. The configuration details are in Table 2:

Parameter Descriptions

The query parameters used in the retrieval process are described below:

- **q:** The main query string, representing the user's information need (e.g., `"Cognitive speed"`).
- **qf:** Field-specific boosting parameters. Assigns weights to fields based on their relevance to the query.
- **pf:** Phrase boosting parameters. Boosts phrases matching specific fields.
- **ps:** Phrase slop. Allows up to 2 words of separation between terms in phrase queries, improving flexibility.
- **ps2:** Phrase slop for longer phrases. Allows up to 1 word of separation for these queries.
- **wt:** The response format for the results. The `json` format is used for easy parsing and processing.

Table 2. Query Parameters for Solr Retrieval

Parameter	Value
q	`Cognitive speed`
qf	`description^3 symptoms^2 causes^2 treatment^1.7 diagnosis^1.5 prevention^1.0 epidemiology^1.5 content^0.5 description^4`
pf	`symptoms^2 causes^2`
ps	`2`
ps2	`1`
wt	`json`
rows	`25`
fl	`name, link, description, symptoms, epidemiology`

- **rows:** Specifies the maximum number of results to return. In this case, the system retrieves up to 25 documents.
- **fl:** Specifies the fields to return in the results.

The queries were sent to the configured Solr endpoints, and the results were evaluated based on the relevance of retrieved documents.

6 Evaluation

Evaluation plays a critical role in information retrieval, heavily influenced by the target document collection and the type of information sought. Understanding potential user scenarios is essential for shaping new designs and implementations, informed by the feedback received. In this project, while doing the evaluation, we focused on the effectiveness of the system's ability to retrieve relevant information and didn't evaluate retrieval speed. For this evaluation, metrics were used that will be explained next.

To make evaluation fair and without putting bias on the more complex systems (with better schema) we will evaluate systems with the same query parameters with a set of metrics grounded on **precision** and **recall** [7] such as **Average Precision (AvP)**, **Precision at K (P@K)**, **Precision-Recall curves**, and **Mean Average Precision (MAP)** were utilized. Precision denotes the percentage of documents pulled that are truly relevant, while recall makes this comparison within all those relevant documents available in the system. As there are over 600 unique documents precise calculation is impractical, however even if precise calculation is not possible, a manual estimation based on the first twenty-five of the returned documents and sampling of others will be a good approximation.

The **Average Precision (AvP)** is definitely one of the most useful as well as crucial measures. It is a well-known fact that the majority of users satisfaction is determined by their precision. In fact, most of the users do not need high recall, as the ratio of the relevant documents retrieved from all the important documents within the system is usually unknown. In **Precision at K (P@K)**, the choice was to evaluate the first twenty-five documents returned per query. This is considered a fair figure that aptly encounters the purposes of a search engine in practice.

The **Precision-Recall Curves** are constructed for each query with direct comparison of systems on the subset of ranked documents returned. In general terms, a system is 'more stable' the smoother the curve formed and is considered better the bigger the precision-recall Area Under the Curve is. This curve shows us the overall effectiveness of the system in balancing precision and recall across different thresholds.

The **Mean Average Precision (MAP)** is a commonly employed measure in information retrieval that provides an average of the Average precision metric through a number of returned sets sustained through an evaluation period. This metric is useful when looking into whether or not the system is robust even when different information needs are put into focus.

In the next parts of this section, possible user queries are presented and evaluated with the aforementioned metrics.

6.1 Symptoms

Query: Cognitive speed.

The information needed for this query was to find disorders that mainly affect logic and cognitive speed. Based on that information, documents that mentioned that the disorder affects the cognitive abilities of a person in the symptom section were deemed relevant. From table 3 we can see that both tasks performed similarly, but not very well since this is a pretty simple query. The problem occurred since speed is a pretty common word even out of this context. When looking at the area under the curves in figure17, the complex system looks like a better performer.

Table 3. Q1 results

Rank	Syst. Simple	Syst. Complex
AvP	0.64	0.67
P@20	0.56	0.6

6.2 Cause

Query: childhood trauma.

The information needed for this query was to find disorders that are often brought to the surface by childhood trauma. Based on that information, documents that in the cause or description part mentioned traumas (especially childhood ones) were deemed relevant. From table 4 we can see that both tasks performed similarly again, with even worse metrics compared to the first query. This could be

attributed to the fact that the system didn't return a lot of relevant documents. Even with that in mind, from figure 18 we can see that most relevant documents are at the top, which is good (later precision falls off).

Table 4. Q2 results

Rank	Syst. Simple	Syst. Complex
AvP	0.6	0.61
P@20	0.44	0.44

6.3 Treatment

Query: Improvement with behavioral therapies

The information needed for this query was to find disorders for which behavioral therapies are effective treatment options. Documents that mentioned behavioral interventions or therapies in the treatment section were consider relevant. From Table 5, we observe that the simple schema slightly outperformed the complex schema, with an Average Precision (AvP) of 0.64 compared to 0.61. Additionally, Precision at 25 (P@25) results indicate that the simple schema returned more relevant documents in the top results (0.68 vs. 0.6). The complex schema, despite being designed for enhanced retrieval, may have been affected by over-filtering or stricter matching criteria. As shown in Figure 19, both systems performed similarly across the precision-recall curve, but the simple schema exhibited slightly higher precision at the top ranks.

Table 5. Q3 results

Rank	Syst. Simple	Syst. Complex
AvP	0.64	0.61
P@25	0.68	0.6

6.4 Pediatric

Query: Frequent on children

The information needed for this query was to identify disorders commonly observed in pediatric populations. Documents that included terms such as "child," "childhood," or "pediatric" in the description, symptoms, or epidemiology sections were considered relevant. From Table 6, we observe that the complex schema significantly outperformed the simple schema in this task. The Average Precision (AvP) for the complex schema was 0.74 compared to 0.41 for the simple schema, and Precision at 25 (P@25) was also higher for the complex schema (0.72 vs. 0.48). This substantial improvement can be attributed to the complex schema's use of synonym expansion and advanced filters, which better matched documents with varied terminology related to childhood disorders. Figure 20 highlights this performance difference, showing that the complex schema maintained higher precision across the recall spectrum. This result demonstrates the value of more retrieval techniques when dealing with nuanced queries like identifying disorders frequently diagnosed in children.

Table 6. Q4 results

Rank	Syst. Simple	Syst. Complex
AvP	0.41	0.74
P@25	0.48	0.72

6.5 Cause with more keywords

Query: caused by genetics inherited.

The information needed for which purpose this query was written was finding disorders that are inherited and can be often seen run in the family. Based on that need, documents that mentioned inherited (hereditary) aspects were deemed relevant. Even though disorders that are mostly caused by mutations in genes without inherent factors weren't seen as relevant, this query was still the most successful of the bunch. That can mostly be attributed to the phrase slop attribute, which gave a better score to relevant documents that had query keywords a little bit "out of place." Precise evaluation metrics can be seen in table 7 and figure 21

Table 7. Q5 results

Rank	Syst. Simple	Syst. Complex
AvP	0.81	0.94
P@25	0.8	0.88

6.6 MAP

Taking into account all the results from the multiple information needs across queries, in the table 8 Mean Average Precision for both systems can be seen:

Table 8. Overall systems evaluation

Global	Syst. Simple	Syst. Complex
MAP	0.62	0.714

Thus, it is concluded that the system demonstrates satisfactory performance. As expected, the more complex system generally produces better results compared to the simpler one.

7 Conclusion and future work

This project demonstrates the implementation and evaluation of an Information Retrieval (IR) system tailored for mental health-related documents using Apache Solr.

The system successfully indexed and retrieved information on mental health disorders by using advanced indexing techniques, including custom analyzers, synonym expansion.

A comparative evaluation of two schemas, simple and complex, highlighted the advantages of sophisticated retrieval methods in handling refined queries.

The results, measured through metrics like Average Precision (AvP), Precision at K (P@K), and Mean Average Precision (MAP), indicate that the complex schema generally outperforms the simple schema, particularly for intricate queries requiring semantic understanding or synonym handling.

However, there are instances where the simpler schema got competitive results, suggesting the need for balanced optimization to avoid over-filtering. Overall, the project illustrates the importance of schema design and query configuration in developing effective search systems.

For future work (Milestone 3), the focus will be on developing the final version of the search system. We plan to develop a user interface to allow interaction with the system, enabling users to explore and retrieve relevant information more effectively. Additionally, efforts will be made to enhance the search engine by incorporating semantic search techniques. This enhanced search engine will help improve the accessibility and quality of the available information about mental and neurological disorders.

8 Search System (Milestone 3)

The previous stage of the project introduced a basic version of the search system. In this phase, we worked on enhancing the system by exploring new features to address these gaps. The main highlights are:

- **Semantic Search**: Using dense vectors [3] to facilitate semantic search by matching documents based on contextual meaning rather than exact keyword matches, enhancing search accuracy and relevance.
- **Rocchio Algorithm**: Improving the relevance of search results by refining them based on feedback or query adjustments.

To evaluate these changes, we followed the same rules and methodology used in the previous milestone. This ensured consistency in testing and allowed us to compare the effectiveness of the new features with the initial version of the system.

For the analysis, we compared two systems: the best-performing system from Milestone 2 and the newly developed embedding-based approaches created for Milestone 3. This comparison allowed us to clearly assess the improvements introduced in this phase.

We used the same information needs from the previous milestone to maintain consistency in our evaluations. By applying these well-defined queries, we ensured that the results could be directly compared, highlighting the improvements made by the new embedding-based system in M3.

We also explored two variations of the Rocchio Algorithm, *User Feedback Rocchio* and *Pseudo-Feedback Rocchio*, and introduced them into our system to refine search results dynamically. These variations allowed us to experiment with different query refinement approaches, optimizing retrieved documents' relevance based on distinct feedback mechanisms.

8.1 Semantic System

Since Solr added support for the storage of dense vectors via the *DenseVectorField* type and dense vector matching through the 'Knn Query Parser", we needed to create first, using a deep learning model ('all-MiniLM-L6-v2' from SentenceTransformer), embeddings for our documents.

For this task, we embedded all the fields that contained important text (title, content, causes, symptoms, ...) in only one field.

For the schema, we didn't change much, we migrated from the previous one, added that *DenseVectorField* to store the vector, and used the same schema for all features we explored. There was two different type of queries for this milestone, a simpler and a more complex one.

8.1.1 Simpler semantic query. This query, table 9, consisted on embedding the query into a dense vector field, using the same model as the one chosen for the documents, after this, using KNN, we retrieved the 25 nearest neighbors of the query vector and the documents vector.

Table 9. Query Parameters for Solr simpler semantic query

Parameter	Value
q	`{!knn f=vector topK=25}{embedding}`
fl	`name,link,score`

8.1.2 Hybrid lexical+semantic query. The hybrid query integrates the lexical search used in M2 with semantic search to provide a robust and comprehensive document retrieval mechanism. Using the *edismax* query parser, the query blends keyword-based matching with dense vector-based similarity to leverage both exact term matching and contextual understanding.

The lexical component is defined using field-specific boosting (**qf** and **pf** parameters), which prioritizes matches based on the significance of fields such as description, symptoms, and causes. The semantic component utilizes dense vector embeddings by embedding the query and leveraging K-Nearest Neighbors (KNN) for relevance matching. This is

reflected in the **bq** (Boost Query) parameter, where the KNN result is treated as an additional boost factor, enhancing the scoring of documents that are both semantically similar and relevant to the lexical query.

By combining these approaches, the hybrid query achieves a balance between the best of the two worlds, capturing "exact" matches while also incorporating meaning-based relationships. Table 10 illustrates the parameters used for this query.

Table 10. Query Parameters for Hybrid Lexical + Semantic Query

Parameter	Value
defType	`edismax`
q	`{query}^0.3`
bq	`{!knn f=vector}{embedding}`
qf	`description^3 symptoms^2 causes^2 treatment^1.7 diagnosis^1.5 prevention^1.0 epidemiology^1.5 content^0.5`
pf	`description^4 symptoms^2 causes^2`
fl	`name,link,score`
rows	25
wt	`json`
ps	2
ps2	1

8.2 Rocchio

For this implementation, we refer to [6] as our reference.

The **User Feedback Rocchio** algorithm updates the query vector based on explicit feedback from users, classifying documents as relevant or non-relevant. By adjusting the weights for the original query, relevant documents, and non-relevant documents (controlled through parameters α, β, and γ), this method ensures that the system adapts to user preferences more effectively. The updated query vector is calculated using centroids for both relevant and non-relevant document sets, moving the query closer to the “ideal” vector.

The **Pseudo-Feedback Rocchio** algorithm, on the other hand, assumes that the top k retrieved documents from the initial query are likely to be relevant. By leveraging this assumption, the query vector is refined using the centroid of these top-ranked documents, with weights adjusted through parameters α and β. This approach eliminates the need for explicit user feedback while still enhancing search relevance through automatic feedback mechanisms.

Both algorithms were implemented using vector operations to adjust the query vectors dynamically. The choice of parameters ($\alpha = 1$, $\beta = 1$, $\gamma = 0.5$ for User Feedback Rocchio, and $\alpha = 1$, $\beta = 0.75$, $k = 4$ for Pseudo-Feedback Rocchio) was informed by iterative testing to balance precision and recall. This dual approach enabled us to evaluate the impact of feedback, both explicit and assumed, on improving query results. While evaluating, user feedback was achieved manually by us with respect to evaluations made on semantic schema before.

9 Evaluation of Improvements

This section presents the results of our evaluation using five distinct queries. For each query, we analyze the performance of our best lexical system, semantic search system and compare the results of different approaches: the basic semantic search, hybrid lexical+semantic search, and the Rocchio algorithm variations. Figures 3, 4, and 5 show the average precision (AvP), P@25, and mean average precision (MAP), respectively, for each search method.

9.1 Symptoms

Query: Cognitive speed.

The information needed for this query was to find disorders that mainly affect logic and cognitive speed. Based on that information, documents that mentioned that the disorder affects the cognitive abilities of a person in the symptom section were deemed relevant. The best retival for this query was achieved with relevance feedback in both metrics, AvP and P@K, which can be seen in Figures 3 and 4 in the Q1 column. The precision-recall curve can be seen in Figure 22 where relevance feedback has the biggest area under the curve.

9.2 Cause

Query: childhood trauma.

The information needed for this query was to find disorders that are often brought to the surface by childhood trauma. Based on that information, documents that in the cause or description part mentioned traumas (especially childhood ones) were deemed relevant. Pseudo-feedback Rocchio showed particular effectiveness here, improving the initial results by incorporating information from top-ranked documents.AvP and P@K metrics can be seen in Figures 3 and 4 in the Q2 column, and the precision-recall curve can be seen in Figure 23, where hybrid search showed poor results compared to other approaches.

9.3 Treatment

Query: Improvement with behavioral therapies

The information needed for this query was to find disorders for which behavioral therapies are effective treatment options. Documents that mentioned behavioral interventions or therapies in the treatment section were consider relevant.Precision-recall curve in Figure 24 shows that relevance feedback clearly outperforms other aproches in this task.Avp and P@K metrics also point to that (Figures 3 and

4) with much difference between AvP value 0.89 for relevance feedback and second best of 0.69 (which was pseudo-relevance aproach)

9.4 Pediatric

Query: Frequent on children

The information needed for this query was to identify disorders commonly observed in pediatric populations. Documents that included terms such as "child," "childhood," or "pediatric" in the description, symptoms, or epidemiology sections were considered relevant. Even though the precision-recall curve in Figure 25 again shows that relevance feedback clearly outperforms other aproches in this task, it is interesting to point out that hybrid search folowed relevace-search curve the longest but then experienced heavy fall in precision as relevance incresed.From Avp and P@K metrics (Figures 3 and 4) same conclusion can be pointed out. Relevance feedback again had a high AvP of 0.95, which outshined other approaches.

9.5 Cause with more keywords

Query: caused by genetics inherited.

The information needed for which purpose this query was written was finding disorders that are inherited and can be often seen run in the family. Based on that need, documents that mentioned inherited (hereditary) aspects were deemed relevant. In this task best precision-recall curve had lexical schema from milestone 2 (with relevance feedback following it closely), which can be seen in Figure 26.From Avp and P@K metrics (Figures 3 and 4) same conclusion can be pointed out.Compared to other tasks, performance on this one was good across all search approaches.

9.6 Overall Performance Metrics

In this section, we present the overall performance metrics across all search approaches. Figures 3, 4, and 5 show the average precision (AvP), P@25, and mean average precision (MAP), respectively, for each search method.

	Q1	Q2	Q3	Q4	Q5
better lexical schema	0.67	0.61	0.61	0.74	0.94
semantic schema	0.75	0.84	0.63	0.75	0.83
semantic (with pseudo-rocchio) schema	0.81	0.88	0.69	0.78	0.80
semantic (with rocchio) schema	0.86	0.81	0.89	0.95	0.91
hybrid (with pseudo-rocchio) schema	0.77	0.41	0.68	0.78	0.83

Figure 3. Average Precision (AvP) for Each Query Across Search Approaches

The AvP results demonstrate the relative performance of each approach in our test queries, highlighting the effectiveness of our different search strategies for specific types of disorder information retrieval tasks.

	Q1	Q2	Q3	Q4	Q5
better lexical schema	0.60	0.44	0.60	0.72	0.88
semantic schema	0.56	0.68	0.60	0.64	0.76
semantic (with pseudo-rocchio) schema	0.60	0.72	0.52	0.68	0.68
semantic (with rocchio) schema	0.72	0.60	0.76	0.84	0.84
hybrid (with pseudo-rocchio) schema	0.68	0.20	0.76	0.48	0.80

Figure 4. Precision at 25 (P@25) for Each Query Across Search Approaches

The P@25 metrics provide insight into the immediate utility of our search results, showing how well each approach performs in terms of relevant documents among the top 25 results.

	Mean Average Precision (MAP)
better lexical schema	0.714
semantic schema	0.760
semantic (with pseudo-rocchio) schema	0.792
semantic (with rocchio) schema	0.884
hybrid (with pseudo-rocchio) schema	0.694

Figure 5. Mean Average Precision (MAP) Across Search Approaches

The MAP scores offer a comprehensive view of each approach's overall performance, taking into account both precision and recall across all queries.It can clearly be seen from the data that semantic search with relevance feedback implemented with the Rocchio algorithm outperformed other search approaches. We also presented the average precision-recall curve comparison (all queries included) in Figure 27 which even more highlights difference in performance between relevance-feedback and all other approaches, with relevance-feedback having most area under the curve and most stable curve as well, which shows stable drop in precision with an increase of retrival

10 Search User Interface

The user interface (UI) was developed using Flask, a lightweight web framework for Python. This interface serves as a bridge between users and the information retrieval system, allowing easy interaction with the search functionalities. The key components of the UI implementation are as follows:

1. **Backend Logic**: The main application logic is defined in `app.py`. Flask handles HTTP requests and responses, enabling interaction between the user and the backend Solr-powered retrieval system. The interface integrates with multiple Solr cores, each representing different schemas or search configurations:
   - **disorders**: Simple lexical schema.
   - **disorders02**: Semantic schema.

- **disorders03**: Hybrid schema.

2. **Embedding Integration**: A SentenceTransformer model (`all-MiniLM-L6-v2`) is loaded in the backend to generate dense embeddings for user queries. These embeddings are used for semantic search tasks, ensuring that user queries are matched with documents based on contextual meaning.
3. **Template Rendering**: The HTML templates, such as `index.html`, are stored in the `templates` directory. `index.html` serves as the main interface, presenting a search bar for user queries and displaying the retrieved results in a structured format. Figures 6, 7, and 8 showcase the interface's design, including the main search view and dropdown menus for selecting search modes and Solr cores.

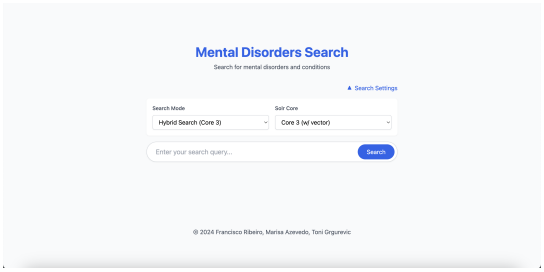


Figure 6. Interface Main Page

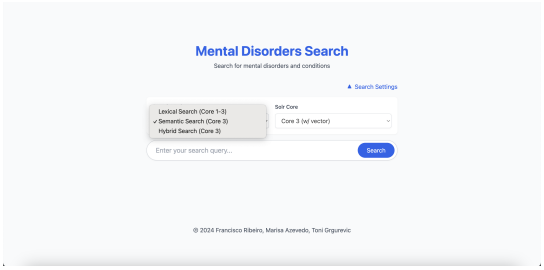


Figure 7. Interface Mode Selection

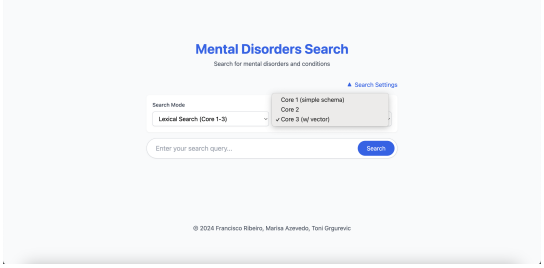


Figure 8. Interface Core Selection

This UI enhances the usability of the search system by providing an accessible and user-friendly platform for interacting with the advanced retrieval functionalities. Future work includes integrating dynamic Rocchio-based query refinement into the interface to further enhance its capabilities.

11 Final System Characterization

The final system integrates the advancements from previous milestones, showcasing significant enhancements in search capabilities and overall performance. The key components of the system are as follows:

1. **Semantic Search Integration**: The system now incorporates a semantic search mechanism using dense vector embeddings , which facilitates contextual matching rather than relying solely on exact keyword matches. This feature improves the relevance of retrieved results, particularly for nuanced and complex queries.
2. **Rocchio Algorithm for Query Refinement**: Two variations of the Rocchio Algorithm were implemented: User Feedback Rocchio and Pseudo-Feedback Rocchio. These algorithms enable dynamic query refinement based on explicit user feedback or inferred relevance from top-ranked results. This enhancement ensures the system adapts effectively to user needs, further improving retrieval accuracy.
3. **Performance Evaluation and Metrics**:
   - The performance evaluation highlights improved Mean Average Precision (MAP) and Precision at K (P@K) values for the semantic and hybrid retrieval systems compared to the baseline schema.
   - Precision-Recall curves demonstrate greater stability and robustness in the system's ability to retrieve relevant documents.
4. **User Interface**: While not essential for system functionality, a user interface was developed to provide an accessible and user-friendly interaction point. This interface allows users to input queries, view retrieved results, and explore advanced search options, enhancing the system's usability.

The final system represents a comprehensive information retrieval solution, combining advanced search algorithms with an optional interface for streamlined user interaction. These improvements solidify the system's capability to support mental health research and analysis effectively.

12 Conclusion and Future Work

Conclusion

The project successfully implemented and evaluated an advanced information retrieval system tailored for mental health-related documents. Significant improvements were achieved through semantic search integration, Rocchio-based query refinement, and detailed performance evaluations. The

results highlighted the robustness and adaptability of the system for various query types.

However, we were expecting better results for the hybrid lexical+semantic query, which did not perform as well as anticipated, particularly in cases where the combination of keyword and semantic matching should have provided a balanced retrieval strategy. This indicates the need for further refinement of the hybrid approach and its parameters.

Overall, the system demonstrates the potential of advanced retrieval techniques in enhancing the accessibility of mental health information, laying a strong foundation for future enhancements.

Future Work

Future work will focus on the following areas:

1. **Improved User Interface**: Enhancing the user interface to allow direct interaction with Rocchio-based query refinement methods. This will enable users to dynamically adjust queries based on their preferences and improve retrieval outcomes.
2. **Hybrid Query Optimization**: Investigating the integration of user feedback-based Rocchio refinements into the hybrid lexical+semantic query. This approach aims to leverage both keyword precision and semantic contextual understanding for improved performance.
3. **Performance Tuning**: Exploring additional techniques to optimize the retrieval parameters and indexing strategies for schemas, particularly for underperforming scenarios like the hybrid queries.

These steps aim to enhance the usability and effectiveness of the system, ensuring it continues to meet the evolving needs of mental health research and information retrieval.

References

[1] Apache Software Foundation. 2024. Apache Solr - edismax. https://solr.apache.org/guide/7_7/the-extended-dismax-query-parser.html Accessed: 2024-11-18.

[2] Apache Software Foundation. 2024. Apache Solr Guide. https://solr.apache.org/guide/solr/latest/index.html Accessed: 2024-11-09.

[3] Apache Software Foundation. 2024. Dense Vector Search. https://solr.apache.org/guide/solr/latest/query-guide/dense-vector-search.html Accessed: 2024-12-06.

[4] Python Software Foundation. 2024. Beautiful Soup. https://pypi.org/project/beautifulsoup4/ Accessed: 2024-10-10.

[5] Toni Grgurevic Francisco Ribeiro, Marisa Azevedo. 2024. GitHub Project Repository. https://github.com/franciscoribeiro2003/Mental_Disorders_SearchEngine Accessed: 2024-12-19.

[6] Victor Lavrenko. 2014. relevance feedback (Rocchio). https://www.youtube.com/watch?v=V6u63kTP9Og Accessed: 2024-12-10.

[7] scikit-learn developers. 2024. Precision-Recall. https://scikit-learn.org/1.5/auto_examples/model_selection/plot_precision_recall.html Accessed: 2024-11-19.

[8] Wikipedia. 2024. Creative Commons Attribution-ShareAlike 4.0 International License. https://en.wikipedia.org/wiki/Wikipedia:Text_of_the_Creative_Commons_Attribution-ShareAlike_4.0_International_License. Accessed: 2024-10-10.

[9] Wikipedia. 2024. Mental Disorders. https://en.wikipedia.org/wiki/List_of_mental_disorders Accessed: 2024-10-10.

[10] Wikipedia. 2024. Neurological Disorders. https://en.wikipedia.org/wiki/List_of_neurological_conditions_and_disorders Accessed: 2024-10-10.

A Appendix

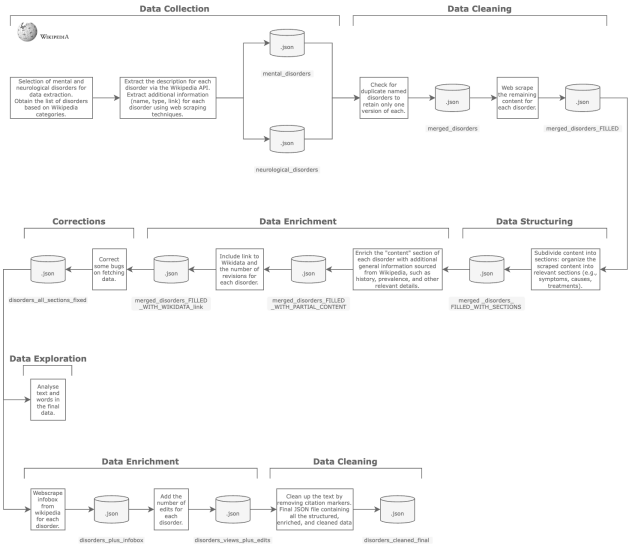


Figure 9. Pipeline Diagram

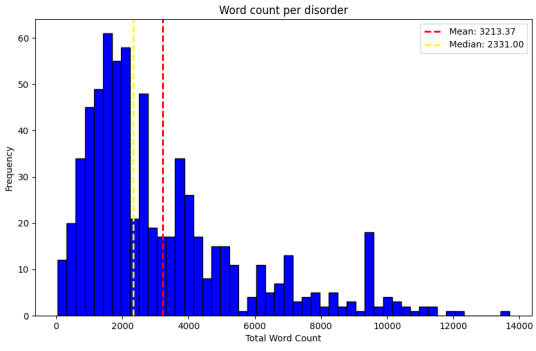


Figure 10. Distribution of document length (number of words) across the dataset

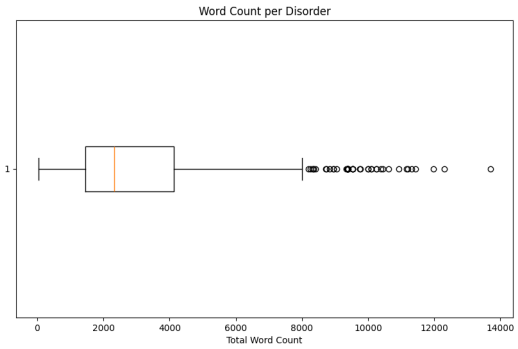


Figure 11. Box plot showing the distribution of document lengths with several outliers

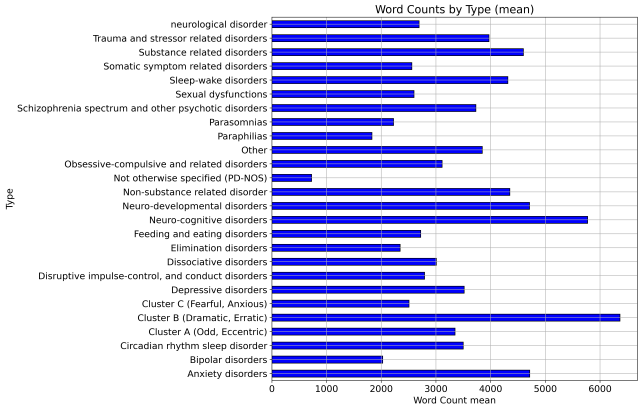


Figure 12. Mean document length by disorder type

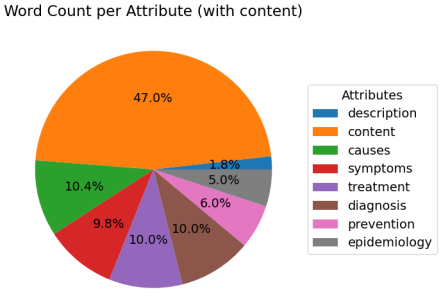


Figure 13. Distribution of document content across different attributes

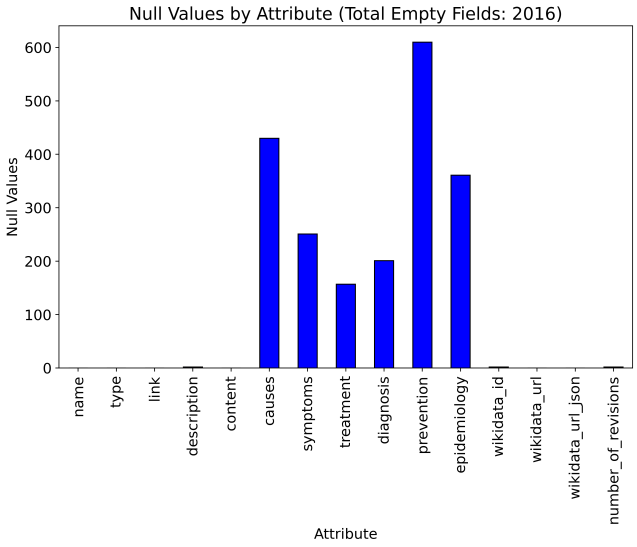


Figure 14. Distribution of null values across different attributes

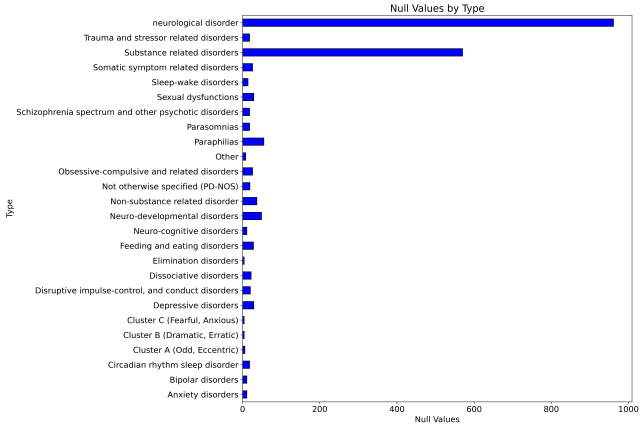


Figure 15. Distribution of null values across different types.

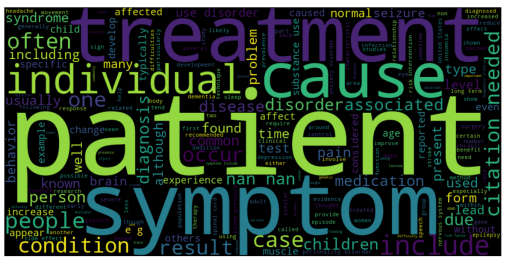


Figure 16. Word cloud representing the most common terms in the dataset.

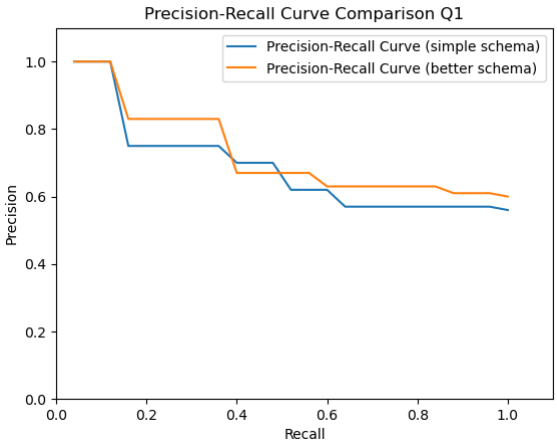


Figure 17. Q1 Precision-recall curve

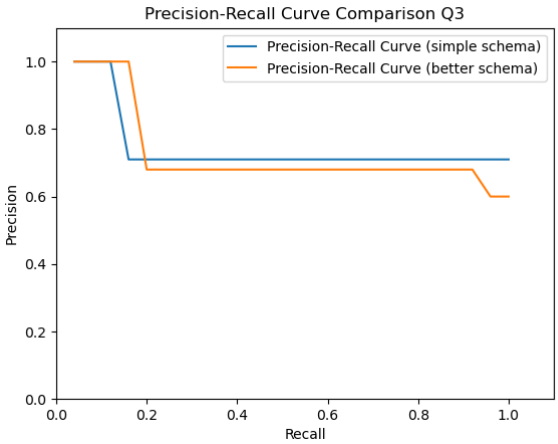


Figure 19. Q3 Precision-recall curve

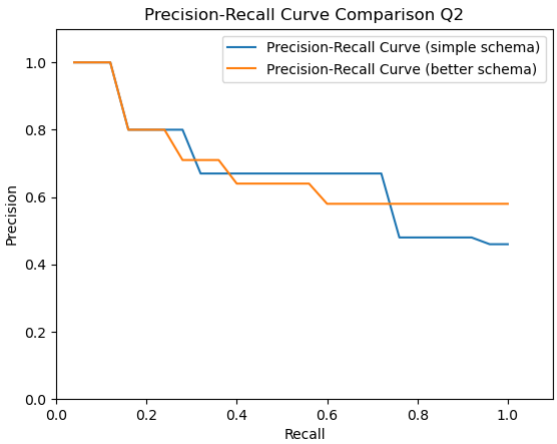


Figure 18. Q2 Precision-recall curve

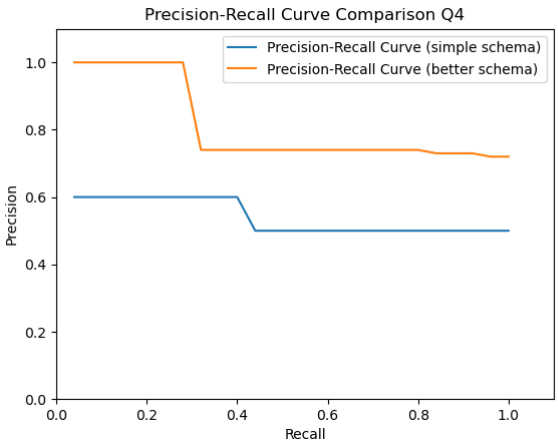


Figure 20. Q4 Precision-recall curve

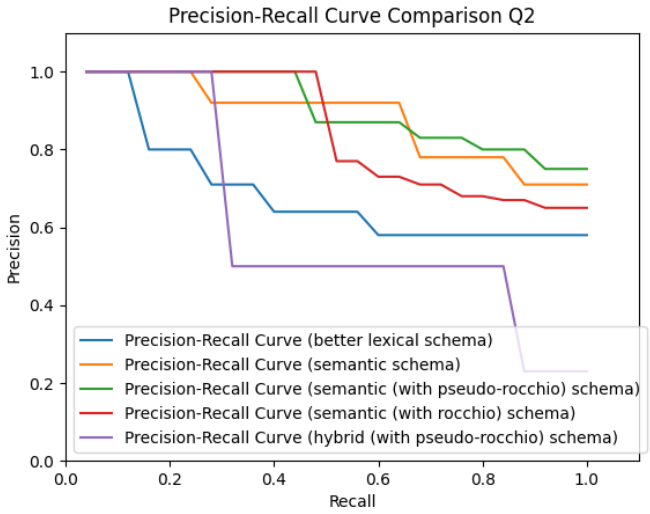


Figure 23. Performance comparison for Query 2 across different search approaches

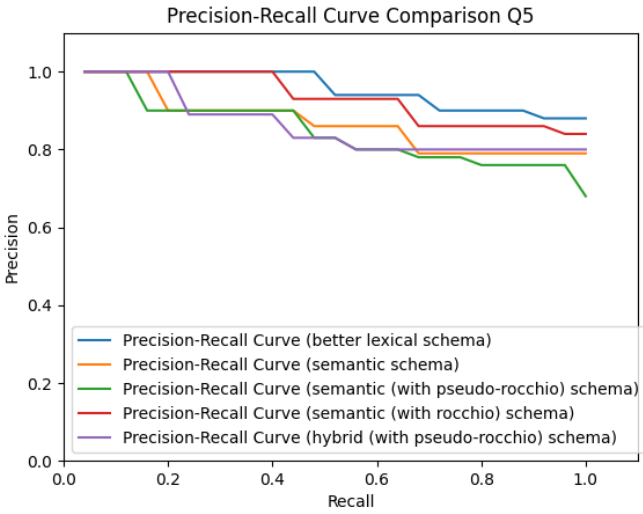


Figure 26. Performance comparison for Query 5 across different search approaches

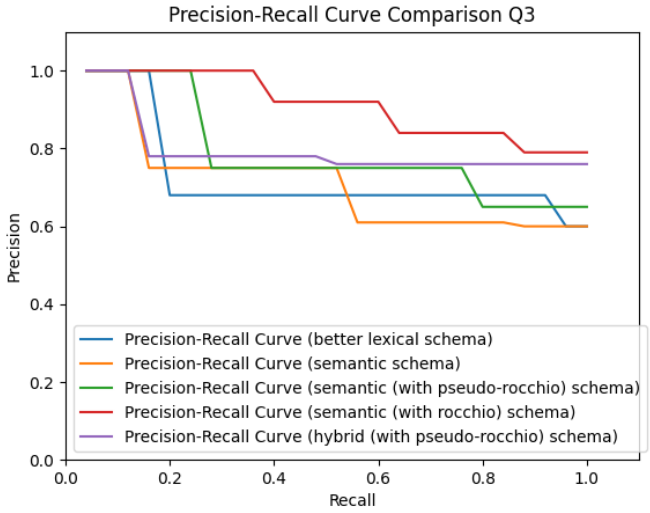


Figure 24. Performance comparison for Query 3 across different search approaches

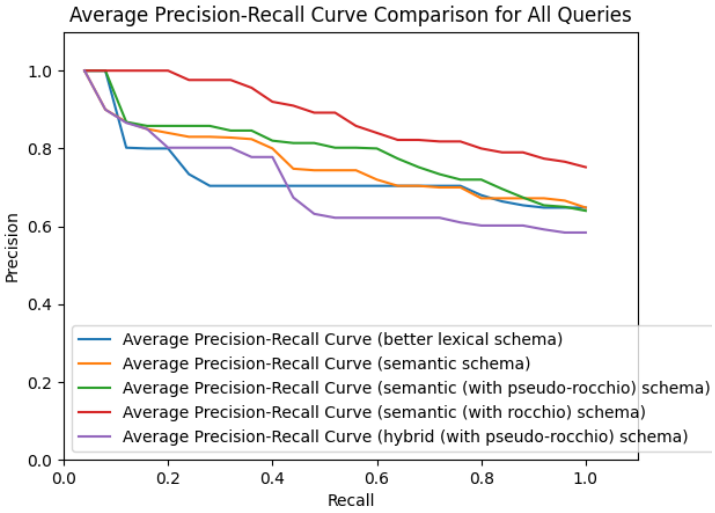
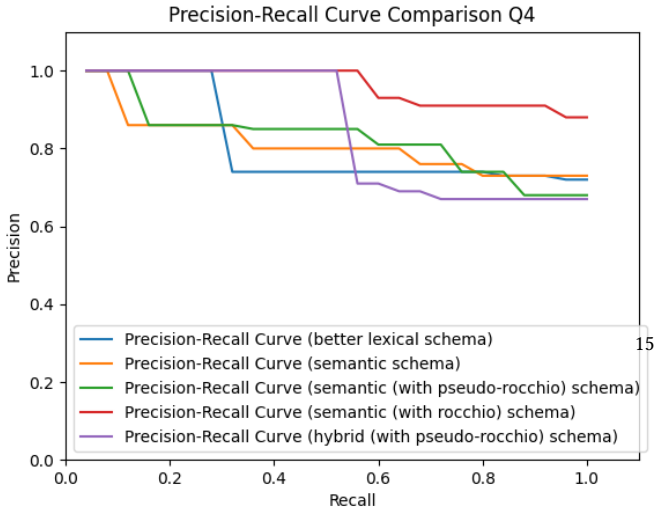


Figure 27. Average performance comparison across different search approaches



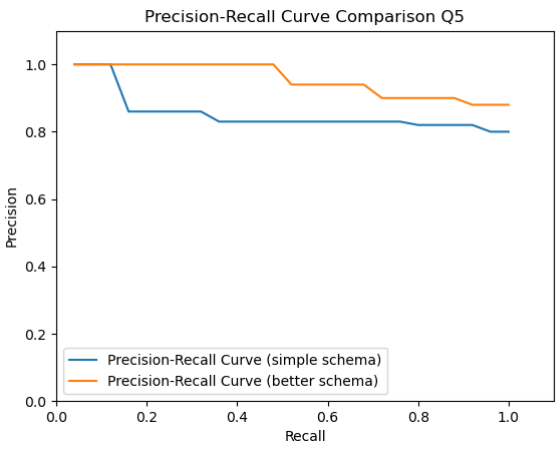


Figure 21. Q5 Precision-recall curve