UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

eXascale Infolab

UNIVERSITY OF FRIBOURG

MASTER THESIS

Towards a Framework for Openness Score Calculation in Scholarly Research

Author:
Jennifer Swaminathan

Supervisor:
Prof. Dr. Philippe Cudré-Mauroux

February 12, 2024

eXascale Infolab
Department of Informatics

Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland
phone +41 (26) 300 84 65 • diuf-secr@unifr.ch • www3.unifr.ch/inf

Abstract

Jennifer Swaminathan

Towards a Framework for Openness Score Calculation in Scholarly Research

This thesis endeavors to establish a comprehensive framework for the creation of an openness metric tailored for scholarly articles. This pursuit is driven by three research questions that lay the foundation for our exploration. Firstly, we delve into a thorough investigation of various scholarly APIs, dissecting their offerings and scrutinizing their potential as data sources. After evaluation, the Openalex API emerges as our primary choice. Subsequently, we examine the contrasting dynamics between utilizing an API and accessing data from an in-house database. This examination unveils intricate trade-offs, ultimately leading to the formulation of a robust workflow. This workflow is engineered to harmoniously integrate API and server capabilities, resulting in the seamless generation of an openness metric provided by visualizations. In our pursuit to enrich these metrics, we harness the power of Natural Language Processing (NLP) techniques, tapping into a wealth of open-access articles. We direct our attention towards question-answering models, seeking to extract valuable insights. To facilitate this process, we curate a dataset comprising 38 articles and derive answers through a singular human response. Our exploration traverses both extractive and abstractive question-answering models, with the latter proving more promising. Among the abstractive models, we look into the GPT series and Llama2, equipped with 7 billion parameters. Additionally, we employ the retrieval augmented generation(RAG) architecture to optimize token usage, coupled with prompts to draw specific contextual responses. Our comprehensive analysis reveals that the GPT models outperform their counterparts in terms of accuracy and response time. While Llama2 exhibits slightly reduced speed, prompt engineering improves its performance. As a conclusion, we present a framework that harnesses the potential of the Llama2 model, with the RAG architecture and strategically tailored prompts. This dynamic combination generates a visualization for the openness score. This approach embraces technologies like abstractive models and NLP and also integrates API and server functionalities, promising a new era in openness assessment.

Keywords: Large language models, Llama2, OpenAI, Question-Answering Models, Retrieval Augmented Generation, Prompt engineering, Big Data, OpenAlex API

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Researchers invest substantial time and resources in conducting experiments, analyzing data, and documenting their findings. However, despite these extensive efforts, a significant portion of this valuable knowledge remains locked behind paywalls and subject to restrictive access policies. A pressing need for a transformative solution is required which is the principles of open science. Central to the concept of open science is the notion of "openness." It encompasses the idea that scientific knowledge should be liberated from constraints.

The Openness Score project, an initiative by the Openness Score team in collaboration with Swiss universities, stands as a significant endeavor in this pursuit. Its mission is to evaluate and enhance the openness of scholarly research, ultimately fostering a culture of openness and transparency within the academic community. Central to this mission is the development of a robust and comprehensive framework for calculating the openness score of scholarly articles.

Partners from Swiss Universities are actively working towards establishing a metric for "Openness." This metric not only assesses the openness of publications but also considers the accessibility of tools, datasets, and databases used in the research process. The initiative aims to create a standardized metric, referred to as the "Openness Score" (OS), which quantitatively measures the openness of scholarly articles.

Open science champions transparency, collaboration, and unrestricted sharing of research findings. To incentivize researchers and institutions to embrace these principles, it is imperative to have a means of quantifying and comparing the level of openness across a wide array of academic articles. The Openness Score seeks to provide a standardized measure for this purpose, empowering stakeholders with a quantitative tool to identify articles that align with their open-access objectives. This enhances the efficiency of information retrieval and utilization.

The Openness Score (OS) serves as a metric, scoring system, or visualization tool designed to assess the degree of openness of a scholarly article. This metric takes into account various factors and data sources to generate a numerical value or visualization that reflects the article's openness. Key considerations in developing the Openness Score include the article's accessibility, whether it is freely available to the public or restricted behind paywalls, subscriptions, or other access barriers. Additionally, it assesses the availability of research tools used during the study for public access.

The development of an openness score metric plays a pivotal role in fostering the adoption of open-access publishing practices among authors, institutions, and publishers. It serves as a benchmark for promoting open science initiatives and practices. This thesis represents a comprehensive analysis aimed at contributing to the

ongoing discourse.

The motivation driving this research unfolds on a dual front: firstly, this study aims to explore the available sources, particularly focusing on Application Programming Interfaces (APIs) within the scholarly publishing domain. We endeavor to discern which APIs hold the potential to provide valuable metadata, and subsequently, how this metadata can be used to formulate a comprehensive openness score. Additionally, we delve into the capacity of certain APIs to enable the download and storage of data within an in-house database on a server. This pursuit facilitates a comparative analysis, shedding light on the disparities between utilizing APIs and a server-based approach in the pursuit of openness metrics.

Secondly, this research delves into the potential of leveraging the textual content within open-access articles to enhance the metric. In this endeavor, we aim to harness the power of artificial intelligence (AI) and natural language processing (NLP) techniques. By subjecting the text of open-access articles to AI and NLP analysis, we seek to identify mentions of datasets and ascertain whether they are openly accessible. These findings could contribute to enhancing the openness score visualization for each article.

In this context, this thesis plays a key role in the examination of components that contribute to the Openness Score metric. Through a detailed analysis, including an evaluation of available APIs, a comparison of server-based solutions, and an exploration of the capabilities of NLPs, this thesis seeks to answer our research questions.

1.2 Research Questions

This thesis addresses three research questions aimed at enhancing our understanding of openness assessment in scholarly publications:

Research Question 1 (RQ1): Which available scholarly APIs are best suited for conducting comprehensive analyses of openness in scholarly publications? We seek to compare and evaluate some of the available scholarly APIs to determine the most suitable one for our analysis.

Research Question 2 (RQ2): In the quest to determine the openness score of scholarly publications, how does the efficiency of API endpoints compare with that of a locally hosted server that contains downloaded content? This comparison is crucial for understanding the most effective and practical approach for assessing the openness of scholarly publications. Efficiency is particularly important in scenarios where users may be submitting queries in real-time, necessitating quick responses to facilitate immediate analysis. By identifying the most efficient method, we aim to optimize the process for real-time user engagement, ensuring that the assessment of scholarly openness is both accurate and timely.

Research Question 3 (RQ3): How effective are Large Language Models (LLMs) in extracting additional information from open-access scholarly articles, which can contribute to the development of a novel openness metric? We explore the potential of LLMs in enriching the openness assessment process by extracting valuable insights from scholarly publications.

To address these research questions, we structure our analysis into the following chapters:

Chapter 2: Related Work provides a review of relevant literature, including topics such as open science, fairness principles, OpenAlex, and other key concepts. This chapter provides the necessary foundation for understanding the core concepts central to our thesis.

Chapter 3: Methodology outlines the methodology used to design and conduct our experiment for each of our research questions.

Chapter 4: Results presents the outcomes derived from our experiments, shedding light on the efficiency of different methods.

Chapter 5: Discussion delves into an in-depth analysis of the findings in the Results chapter.

Chapter 6: Conclusion, offers a comprehensive conclusion to our thesis, addressing the limitations encountered in our research and proposing avenues for future work and research endeavors.

Through this structured approach, we endeavor to provide valuable insights into openness assessment in scholarly publications, contributing to the broader discourse on open science and transparency in research. By developing and sharing a flexible, reproducible framework that adheres to FAIR principles, our study exemplifies open science in practice. We enable the adaptation of various Large Language Models (LLMs) within our framework, demonstrating their utility across different research domains. Furthermore, by openly sharing our findings, we foster a collaborative research environment that encourages further exploration, critique, and validation. This initiative aligns with the principles of open science, emphasizing the importance of accessibility, interoperability, and reusability in the research community.

Chapter 2

Related Work

While developing a novel metric for measuring the openness of scholarly articles, it is essential to delve into the various aspects of open science and related concepts. This section provides an overview of pertinent themes and existing tools and resources that inform the development of such a metric.

2.1 Open Science and Open Access

Open science is a goal aimed at ensuring transparency and accessibility in scientific research. In Sciences, Medicine, et al. (2018), the three primary concepts of open science include open access, open data, and open source. However, in Chakravorty et al. (2022), the authors expand upon these concepts to include open methodology, open resources, and open peer review. Open science has redefined the way research is conducted and disseminated, embodying a commitment to openness that includes practices such as open access to publications, open data sharing, and open-source software. Open science has significantly contributed to scientific progress. Besançon et al. (2021) demonstrate how open science played a vital role during the Covid-19 pandemic, potentially saving millions of lives through knowledge sharing among researchers. By making research outputs, including articles and datasets, freely available to the public, open science democratizes access to knowledge and provides valuable information to the research community. It enables researchers to build upon existing work, reducing duplication of efforts, and accelerating the pace of scientific discovery.

A cornerstone of open science is open access, ensuring that research outputs, particularly articles, are freely available to anyone. This concept transcends traditional paywalls, making research accessible to a global audience. Kidwell et al. Kidwell et al., 2016 discuss how incentives such as providing a badge for open access publication in Psychological Science have increased the number of open research publications. Additionally, a study in McKiernan et al., 2016 analyzes research with open accessibility, revealing higher citation rates compared to closed research."

To measure the openness of an article, it is crucial to consider the extent to which it adheres to the principles of open science. This includes assessing whether an article is openly accessible to the public or locked behind subscription walls. Also, some articles, even if openly accessible could use data that are not openly accessible to the public. Open data can also allow reproducibility and also detect false information as mentioned in Burgelman et al. (2019). While developing a new metric for the openness of a scholarly article, all the information regarding the software and data used in research needs to be taken into account.

2.2 FAIR Principles

An integral part of open science is adhering to the FAIR guiding principles (Wilkinson et al., 2016), which stands for **Findable, Accessible, Interoperable, and Reusable**. These principles provide essential guidance for the management and sharing of data and other research outputs to maximize their utility. In Jacobsen et al. (2020) the authors provide implementations to help researchers follow the FAIR principles effectively. Making research **findable** ensures that it can be discovered by interested parties. **Accessibility** guarantees that data and articles are easily retrievable, promoting transparency. **Interoperability** allows seamless integration of different datasets, while **reusability** encourages the effective utilization of research outputs. As highlighted by Mons et al. (2017), while the research results are typically published, the data must adhere to the FAIR principles. It is important to note that FAIR does not mandate that data must be 'open,' but it does require data to be transparent and clear, enabling the reproducibility of methodologies.

2.3 APIs for Open Access Determination

When devising a metric to assess the openness of scholarly articles, access to relevant metadata is crucial. In this subsection, we explore various Application Programming Interfaces (APIs) that provide valuable metadata related to scholarly articles. These APIs provide information regarding the articles, like the Digital Object Identifier (DOI), title, date of publication, if the article is open access etc. Several APIs have been developed to assist in providing this valuable information. Notable APIs considered in this study include **Crossref**, **Unpaywall**, **DOAJ** (Directory of Open Access Journals), and **OpenAlex**. These APIs offer a means to access the metadata associated with scholarly articles. Visser, Van Eck, and Waltman (2021) perform comparison between five APIs which includes Crossref API as well. they highlight the magnitude of metadata that is available to perform a large-scale analysis.

Crossref (Pentz, 2001) is a widely recognized API that provides comprehensive metadata for academic content, including open access status, and is a vital scholarly data source. Hendricks et al. (2020), summarizes the use of this API, and how the metadata are registered by their members who wish to register their publication with a Digital Object Identifier(DOI).

The **Directory of Open Access Journals (DOAJ)** (Morrison, 2017) is a curated database of open access journals, cataloging articles and their associated metadata. While it serves as an essential resource for open-access publishing, it is important to note that not all open-access articles are listed within it. Thus, a comprehensive approach to determining article openness is essential.

Unpaywall (Dhakal, 2019), conversely, maintains a comprehensive dataset of Open Access articles, making it a valuable resource for assessing the accessibility of articles. Notably, Unpaywall assigns distinct open access statuses to each article, categorizing them as green, gold, hybrid, or bronze. This classification system provides a nuanced understanding of article openness. Unpaywall sources its data from authoritative repositories, including PubMed Central, Crossref, DOAJ, and Data Cite, ensuring the reliability and accuracy of its information.

Microsoft introduced the **Microsoft Academic Graph (MAG)** (Wang et al., 2020), a comprehensive resource for accessing and analyzing research-related data. Although MAG discontinued its API service in 2022, its work has been integrated into **OpenAlex**, making it a valuable source for structured data on scholarly articles.

OpenAlex, as detailed in (Priem, Piwowar, and Orr, 2022), has emerged from the integration of data from MAG, DOAJ, and Crossref. This integration empowers OpenAlex to offer extensive metadata and robust entity recognition capabilities. For researchers seeking comprehensive information about scholarly articles, OpenAlex is an invaluable resource. Notably, it operates as a fully open-source platform for scholarly metadata.

As part of the open science movement, the importance of metadata analysis within scholarly APIs becomes evident. Recent studies, by Rizzetto and Peroni (2023), have highlighted the significance of mapping entities between open bibliographic metadata collections, such as OpenCitations Meta and OpenAlex. These mappings serve to integrate internal identifiers and align bibliographic resources, offering unique insights into the consistency and accuracy of metadata. Furthermore, Jiao, Li, and Fang (2023) shows the challenges posed by inconsistent indexing of data journals and their publications across popular scholarly databases. This inconsistency hampers quantitative analysis efforts in understanding how research data is published and reused. In this context, our thesis aims to perform a comparative analysis of the metadata provided by scholarly APIs. Such analysis is essential to validate and inform the design of new frameworks that leverage these APIs for openness score calculations.

2.4 Natural Language Processing

In recent years, there has been significant innovation in the field of natural language processing (NLP). One of the groundbreaking methods for text-related computations is the use of transformers, a concept introduced by Vaswani et al. in their influential work in 2017 (Vaswani et al., 2017). Transformers have enabled the accomplishment of various text-related tasks, including text translation, summarization, sentiment analysis, text classification, and text generation. An overview of various transformer models can be found in the work by Gillioz et al. (2020).

One significant development in NLP is the emergence of Large Language Models (LLMs), such as BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018), GPT (Generative Pretrained Transformer), and their variants. These LLMs are trained on vast amounts of text data and have achieved remarkable success in various NLP tasks, including question-answering, language translation, and text generation.

Question-answering(QA) models are trained to read a passage of text and provide accurate and contextually relevant answers to questions based on the content of the passage. These models are commonly used in various applications, including information retrieval, virtual assistants, chatbots, and more. They aim to bridge the gap between human language and machine understanding by enabling computers to comprehend and respond to questions in a human-like manner.

QA models, as discussed by Allam and Haggag (2012), encompass Information Retrieval (IR), Information Extraction (IE), and Natural Language Processing (NLP) tasks. They consist of three primary components: Question Processing, Document Processing, and Answer Processing (Mervin, 2013). In our thesis, we compare two extractive QA models, Bert and Roberta (Liu et al., 2019) for our experiments, which extract information from the provided text and provide the answers from it.

In our study, we also study other LLMs, such as the GPT3.5 series and LLama2 (Touvron et al., 2023), which are capable of extracting information from external knowledge sources and generating their responses. The GPT3.5 series has brought

about significant advancements in natural language processing tasks, including translation, content generation, and customer services. These models are trained with numerous parameters and offer APIs that enable us to perform question-answering tasks by providing questions to the models, allowing them to extract information from external knowledge sources. However, they are paid services. In a comprehensive analysis conducted by Ye et al. (2023), a comparative study of the GPT series revealed that despite the series' evolution, there is no clear evidence of improved performance in Natural Language Processing to understand tasks. Additionally, Roumeliotis and Tselikas (2023) provides a detailed list of both the advantages and limitations of these models. On the other hand, Llama2, an open-source language model developed by Meta, offers a range of parameter configurations, including 7, 13, or 70 billion parameters. For our study, we specifically focus on the 7 billion parameter version.

These generative LLMs have limitations on the number of tokens that they can handle and also have issues of hallucinations as discussed by Zhang et al. (2023). Given these limitations of token restrictions and hallucination, we explore the Retrieval Augmented Generation (RAG) model. When we only ask a question about an article, the Large Language Models (LLMs) can hallucinate, but by providing external domain knowledge we can reduce its hallucination as suggested by Chen et al. (2023). Lewis et al. (2020) explore various RAG formulations to reduce hallucinations. In RAG models, a retrieval component is employed to extract the 'K' most pertinent passages from the document, which are then provided to the LLM for answer generation. Additionally, **prompting** techniques, when applied to LLMs, provide fine-tuned control over information extraction, making them valuable tools for this task. Arvidsson and Axell (2023) perform a literature review on prompt engineering, demonstrating its guidelines and discussing its infancy in providing accurate results.

In summary, this section has presented an overview of the key concepts and resources essential for developing a framework to assess the openness of scholarly articles.

Chapter 3

Methodology

In this chapter, we outline the methodology used to address our research questions, which focus on comparative analysis of available APIs, exploring their effectiveness as an alternatives to an in-house data server, and employing Large Language Models (LLMs) to extract additional information for enhancing the Openness Score metric.

Our chapter is structured to systematically address each research question by detailing the data collection and analysis methods employed in our experiments.

3.1 RQ1 : Comparative Analysis on scholarly APIs

When assessing the openness of scholarly articles and deriving a corresponding score or visualization, it is essential to gather information about each article. To accomplish this, we explore various APIs that offer access to this information. Our goal is to select an API that ensures accuracy and reliability in the data it provides. To achieve this, we have chosen to analyze four well-known APIs.

3.1.1 Data collection

The APIs that we have chosen for performing our comparative analysis are.

- Crossref
- Doaj
- Unpaywall
- OpenAlex

Each of these APIs have their endpoints that grant access to their metadata. For our analysis, we primarily rely on the use of Digital Object Identifiers (DOIs) to access information and retrieve comprehensive metadata.

Crossref is a not-for-profit membership organization that works to make scholarly content easy to find, link, cite, and assess. With its REST API[1], Crossref which allows us to obtain the metadata by using its endpoint for extracting the information regarding a DOI. This system plays a crucial role in the scholarly communication ecosystem by providing infrastructure services for DOI registration and metadata dissemination, supporting the traceability and accessibility of academic research. Crossref represents over 17,000 members from 146 countries, managing more than 130 million records and processing over 600 million metadata queries monthly.

[1] https://www.crossref.org/documentation/retrieve-metadata/rest-api/

DOAJ, as a meticulously curated directory of reputable open-access journals, serves as an authoritative source for identifying high-quality open-access publications. It also provides an API [2] that allows us to access metadata for scholarly articles. Managed by Infrastructure Services for Open Access (IS4OA), a non-profit organization, DOAJ's mission is to increase the visibility, accessibility, reputation, usage, and impact of open-access scholarly research.

Unpaywall API [3] is another resource that enables us to access information about various entities, including articles, based on their DOIs. Unpaywall, a project by OurResearch, also a non-profit organization, is dedicated to making scholarly research more open, connected, and reusable. Focusing on peer-reviewed scholarly articles and pre-publication versions from repositories like arXiv, Unpaywall specializes in locating content with DOIs, bypassing traditional paywalls to make scientific knowledge freely accessible. As a non-profit initiative, Unpaywall contributes significantly to dismantling access barriers in scholarly communication, aligning with the broader goals of open science.

OpenAlex API [4] provides a wide range of scholarly entities like authors, works, publishers, etc, making it a very informative source of information. OpenAlex is operated by OurResearch, continuing the commitment to enhance the discoverability and accessibility of scholarly knowledge. OpenAlex's mission is to make the entirety of global research accessible to everyone, anywhere, contributing to a more interconnected and inclusive scholarly ecosystem. Its open-source access to a comprehensive snapshot position it as a strong candidate for our analysis.

3.1.2 Data analysis

With the selected APIs, we conduct a comparative analysis to determine the information available through each of them and the features that can be utilized to calculate the metric defining the openness of an article. During our research, we opt for a comparative analysis of these chosen APIs. To perform an analysis of the selected APIs, we aim to identify differences in the following characteristics.

- Rate limits
- Access Speed
- Public dump availability
- Data coverage and metadata depth

We have selected these specific characteristics because, when developing an application that relies on APIs for calculations, understanding the rate limits is of great importance. Rate limits can impose constraints that must be taken into account during the application's development. This knowledge helps in proper resource allocation, ensuring that API usage remains within acceptable boundaries and preventing overuse that could lead to service disruptions.

Furthermore, rate limits play a crucial role in designing workflows that efficiently utilize API resources. By optimizing data retrieval and analysis processes, we can minimize bottlenecks and enhance the application's performance. Effective management of rate limits can also be instrumental in controlling costs, particularly

[2] https://doaj.org/api/v3/docs

[3] https://unpaywall.org/products/api

[4] https://docs.openalex.org/how-to-use-the-api/api-overview

when dealing with APIs that charge based on usage. Staying within the specified rate limits is essential to avoid unexpected charges. The outcomes of our analysis on rate limits are detailed in Section 4.1.1.

Secondly, access speed is a critical factor. Speed is imperative as it directly impacts the efficiency of our calculations and the overall user experience. Timely access to data is essential, and we measure the time taken to retrieve metadata for various DOIs from the selected APIs to determine the average access speed for obtaining metadata for a single DOI and present in results in Section 4.1.2. Faster data access is particularly important for real-time or time-sensitive research, ensuring that users can access information quickly and efficiently.

We also consider the possibility of downloading metadata, which is crucial as we explore the use of an in-house server for openness metric calculations, which we compare in Section 4.1.3. The ability to store these results for further analysis is a key consideration. Additionally, public dumps of data can serve as a backup or redundancy option, reducing the risk of data loss due to API changes or outages.

Furthermore, we assess the relevance of the information contained in the metadata provided by these APIs for calculating openness scores in Section 4.1.4. We compare the various entities for which the APIs offer information.

Incorporating these analyses into our research ensures that we make informed decisions when selecting APIs, aligning them with our specific research needs and objectives.

3.2 RQ2: Comparing efficiency of API and in-house database

In this section, we outline the methodology employed to address our second research question: the comparative evaluation of utilizing an API versus employing an in-house server with data sourced from the OpenAlex snapshot.

This section first presents the data collection process that was followed. We discuss the purpose for the selection of our sources, following which we elaborate on the data analysis section that we will be considering to perform our experiment.

3.2.1 Data collection

In this section, we discuss the reasons for making our design choices to answer our RQ2. From the analysis performed to answer our research question 1, we find that OpenAlex is a reliable scholarly API that contains 240M records and the possibility to download its data. So we decide to download its content.

To download the snapshot of OpenAlex we procured a dedicated server with the following specifications:

- 4TB SSD for storage
- 32 GB RAM for memory

We obtained a snapshot of the OpenAlex database, which was made available in an AWS S3 bucket. The AWS S3 bucket is a cloud storage service provided by Amazon Web Services(Baron and Kotecha, 2013). It is a scalable and secure platform for storing and retrieving data. In our case, OpenAlex stores its monthly data snapshot in an S3 bucket, making it accessible to users. The OpenAlex snapshot was copied to our dedicated server using the instructions provided in the official documentation[5].

[5] https://docs.openalex.org/download-all-data/openalex-snapshot

The OpenAlex data is stored in the OpenAlex bucket and is gzip-compressed, with each entity represented as one row per file. Data is organized into folders or prefixes based on entity types, such as authors, works, and more. Each entity file is further categorized by its updated date. The manifest file, in JSON format, lists all data files for each entity type. The compressed snapshot is approximately 330 GB in size and expands to about 1.6 TB when decompressed. Downloading the OpenAlex snapshot data is free and does not require an Amazon account. We use the Amazon Web Services Command Line Interface (AWS CLI) to copy the data to our server, taking up about 330 GB of disk space. The data is organized into folders for authors, concepts, institutions, sources, and works, categorized by their updated dates.

Once the data is downloaded to the server, it is converted from JSON format to CSV format. The time required for this conversion may vary, ranging from a few hours to a few days, depending on the network connection and the size of the data. Subsequently, we proceed to load the data into a relational database to facilitate our analysis. We have opted for the PostgreSQL database system (Stonebraker and Kemnitz, 1991), recognizing that this choice plays a pivotal role in determining the overall performance, data integrity, and scalability of our research project.

PostgreSQL offers several advantages that make it a compelling choice for our research:

- **ACID Compliance:** PostgreSQL is known for its strong support of the ACID (Atomicity, Consistency, Isolation, Durability) properties, which ensures data integrity and consistency, critical for research involving scholarly article data.
- **Extensibility:** PostgreSQL allows us to create custom data types, operators, and functions, which is particularly beneficial when working with scholarly data that may have unique attributes and requirements.
- **Advanced Indexing:** PostgreSQL provides a wide range of indexing options, including the B-tree algorithm we use in our research, allowing us to optimize data retrieval and query performance.
- **Community and Ecosystem:** PostgreSQL has a robust and active open-source community, which ensures ongoing support, frequent updates, and a wealth of extensions and plugins that can be leveraged in our research.

By choosing PostgreSQL, we believe we have a reliable and extensible database system that best suits the needs of our research project, providing the necessary foundation for data storage, retrieval, and analysis. Given the substantial volume of Big Data we need to process, including structured data from OpenAlex, our decision to employ a relational database management system (RDBMS) is supported by the findings of the experiment conducted by Jung et al. (2015), which demonstrated its favorable performance characteristics.

Once the data has been successfully converted from JSON to CSV, it is ready for storage in our PostgreSQL database. We use the initial table setup code from [6] to load the initial snapshot into our Postgres database. From the converted data, all the tables are populated with the data. This process took a few days. and now we have a total of 1.6TB of data in our Postgres database to perform our experiment.

The schema is organized into several categories, with a primary focus on the "Works" section, which is crucial for our investigation into the openness score of articles. In total, we have 31 tables, categorized as follows:

[6] https://github.com/ourresearch/openalex-documentation-scripts/blob/main/openalex-pg-schema.sql

The authors category contains 3 tables related to authors and their affiliations. While not the primary focus of our research, these tables are essential for understanding the authorship and collaboration aspects within scholarly works. The concepts category captures information related to concepts, keywords, and subject classifications. These 5 tables help us categorize and analyze the content of scholarly articles. Institution-related tables contain 5 tables that store data about the institutions to which authors are affiliated. This information is valuable for assessing the institutional impact on scholarly works. The publisher category contains 3 tables that record details about the publishers of scholarly articles. This data is crucial for tracking the sources of publications and their influence. The source category contains 3 tables and stores information about the sources of the articles, such as journals, conferences, and publications. Understanding the sources is essential for assessing the quality and impact of scholarly works. The "Works" section is the central focus of our research. Within this category, 12 tables encompass various aspects of scholarly articles. We are particularly interested in these tables to compute the openness score of articles, which is a critical aspect of our research. These tables contain data such as article titles, publication dates, citation counts, URLs of the PDF, and the references of these works.

Each of these tables is designed to capture specific attributes and relationships that enable us to conduct an in-depth analysis of scholarly articles. The schema ensures data integrity, and relationships between tables are defined to support complex queries and data retrieval.

The design of the database schema is tailored to the requirements of our research, allowing us to efficiently manage, analyze, and derive insights from scholarly article data.

For our experiment, ensuring swift data retrieval is crucial. To achieve this, we applied primary indexes to our tables, optimizing data access. Given the scale of the data we handle, careful selection of indexed columns is paramount. In our analysis, we made necessary modifications to the initial OpenAlex schema setup. For instance, in the 'openalex.works' table, we established a primary key on the 'id' column and created indexes on 'DOI' and 'TITLE' for enhanced retrieval speed. In the 'openalex.works_referenced_works' table, we encountered duplicates in the 'work_id' column, as each 'work_id' points to a 'referenced_work_id.' Consequently, we opted to create indexes on these columns to facilitate faster access.

To accommodate future snapshots and updates, we also incorporated conditional alterations to our tables. It is important to note that adding these indexes and primary keys increases storage requirements, as indicated in Table 3.1. Therefore, when provisioning storage resources, it is important to account for the temporary space needed during index creation, even if the final index size is smaller.

To enhance data access speed, we applied indexing to our tables using the B-tree algorithm. This optimization significantly improved query performance and accelerated data retrieval from the database. With the database fully initialized, containing the data and the applied indexes, we are prepared to proceed with our experiments.

3.2.2 Data analysis

To compare the performance between API and server we perform two main analyses, the response time between using API and server for the calculation of the openness score metric and also how accurate are the information that is present in the snapshot.

TABLE 3.1: Index Sizes

Index Name	Total Size
authors_pkey	8498 MB
authors_ids_pkey	5266 MB
concepts_pkey	6968 kB
concepts_ids_pkey	6968 kB
concepts_related_concepts_related_concept_id_idx	13 MB
concepts_related_concepts_concept_id_idx	13 MB
concepts_ancestors_concept_id_idx	12 MB
concepts_pkey	6968 kB
concepts_ids_pkey	6968 kB
institutions_ids_pkey	12 MB
institutions_geo_pkey	12 MB
institutions_pkey	12 MB
publishers_pkey	1192 kB
publishers_ids_pkey	1192 kB
unique_work_concept	185 GB
unique_work_mesh	33 GB
unique_works_primary_locations	30 GB
works_id_ref_idx	29 GB
concepts_work_id	27 GB
works_biblio_work_id_unique	26 GB
biblio_works_id	26 GB
unique_author_counts_year	26 GB
works_title	24 GB
works_locations_work_id_idx	18 GB
works_primary_locations_work_id_idx	16 GB
works_pkey	13 GB
works_unique	13 GB
works_open_access_unique	13 GB
works_open_access_pkey	13 GB
works_ids_unique	13 GB
works_doi_idx	10141 MB
unique_works_best_oa_locations	4200 MB
works_best_oa_locations_work_id_idx	3457 MB
unique_institution_year	108 MB
unique_concept_related_concept	91 MB
unique_concept_year	88 MB
sources_ids_pkey	25 MB
sources_pkey	25 MB

Response time

To compare the performance between the API and server, we built a Flask application in Python. In developing our application, we followed the workflow shown in Fig 3.1 to conduct our experiment. We input a DOI or the title of an article, which is then provided to both the API and the PostgreSQL database. The application first verifies the validity of the DOI or title in the API or the database. If it is not present, the process does not proceed to calculate the Openness score metric. However, if the DOI or title is valid, the calculation of the Openness score metric is performed.

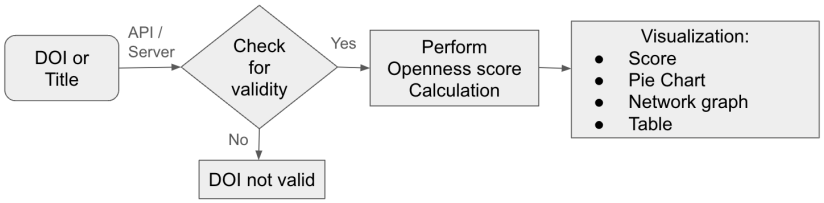

FIGURE 3.1: Workflow of Openness score calculation

Visualisation: *Score*: In the experiment, to determine the openness score of an article, we employ the formula shown in Equation 3.1, as provided by our Swiss university partners. According to the team responsible for the score calculation, a crucial characteristic in assessing the openness of an article is the extent to which its references are open. Articles with a higher number of open references receive a better score compared to those with fewer open references.

$$OpennessScore = (NDOIrOA * 10/NDOIr) * K \tag{3.1}$$

where K = 1 , NDOIrOA is the number of DOIs that are open-access, NDOIr is the total number of references of the given DOI.

Pie Chart: While Equation 3.1 does not assign weightage to the article under consideration, our Openness score team also suggests a visualization method using a nested pie chart to represent these findings. Examples of the desired pie chart are shown in Figure 3.2. In this visualization, the inner pie represents the open access status of the article in question, with open status represented in blue and closed status in yellow. Initially, the colors chosen were green and red for openness, but after conducting an initial survey and considering feedback from peers, it was found that red and green posed challenges. Therefore, in this thesis, we decided to use blue and yellow. The outer pie chart represents the open-access status of the references cited in the article, where open-access articles are depicted in blue and closed-access articles in yellow.

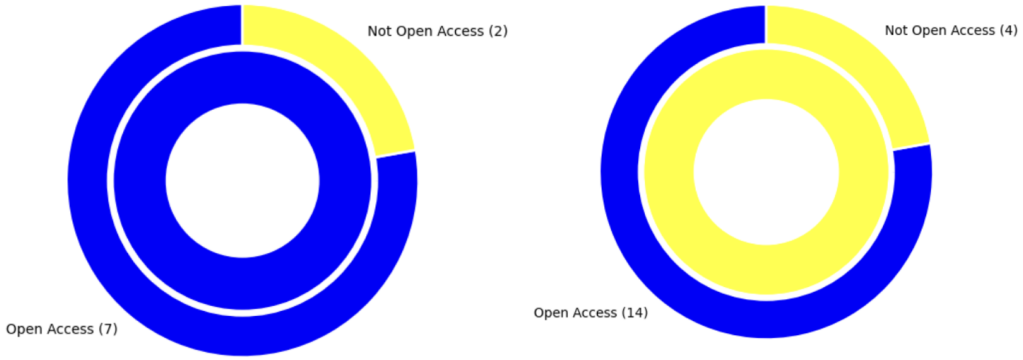

FIGURE 3.2: Openness Score visualization:Pie Chart

In addition to the nested pie chart visualization, we have opted for a network graph to convey the relationships between articles, as illustrated in Figure 3.2. In this graph, the central node signifies the main article, while the connected nodes represent its references. Articles designated as open access are rendered in blue, while those that are not are shown in yellow. The connections in the graph are denoted as follows:

When both the main article and the reference are open access, the connection is represented by a blue line. Conversely, if neither is open access, the connection is indicated by a yellow line. In cases where either the main article or the reference is not open access, the connection is depicted in orange.

This graphical representation effectively captures the flow of articles and their interconnections. As a potential avenue for future research, exploring the references of the references could offer deeper insights into the intricate network and relationships among scholarly articles.

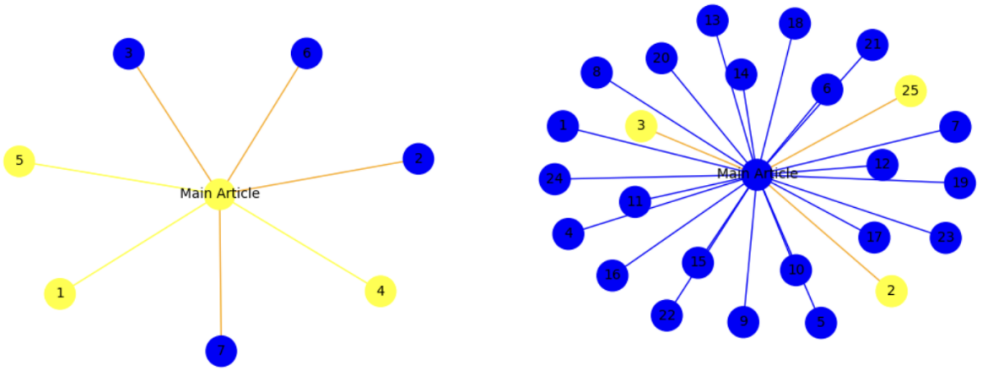

FIGURE 3.3: Openness Score visualization: Network graph

Table: Another way to visualize the openness of an article is through a table that displays its references. In this table, we include details such as the article names, their open access status, and the associated licenses.

These visualization techniques are essential for our analysis, offering an initial insight into the possibility of collecting and presenting fundamental information to users based on the data obtained from the OpenAlex snapshot about the articles.

Experiment: To conduct our analysis, we randomly selected 100 articles, consisting of 50 DOIs and 50 titles, to determine the time required to retrieve information using both the API and the server.

We followed the workflow shown in Figure 3.1 to execute our steps and timed the entire calculation process. This experiment was repeated 10 times to obtain access speed data and the results are provided in Section 4.2.1.

Evaluating data consistency

APIs often provide real-time or frequent data updates, making them invaluable for researchers seeking current and accurate records. This currency ensures that research remains relevant and credible. On the other hand, in-house servers may have less frequent updates, potentially leading to outdated information. By comparing data between the API and the server in Section 4.2.2, researchers can gauge the extent of this difference, ensuring that their work is based on the most recent data available. This comparison also highlights the advantages of APIs in terms of data currency, underscoring their importance in modern research endeavors.

Furthermore, assessing the data discrepancy between an API and an in-house server aids researchers and organizations in making informed decisions about resource allocation. If an API consistently offers more current data, organizations may opt to prioritize API usage to gain a competitive edge in research. This decision-making process relies on understanding the practical implications of data differences

and the potential consequences of relying solely on in-house servers with less frequent updates. For this purpose, we perform analysis on the amount of records that are present in the API and the snapshot for a comparison. In summary, comparing data records between APIs and servers is pivotal for research relevance, decision-making, and resource optimization in today's data-driven world.

3.3 RQ3: Evaluation of LLMs on scholarly articles

In recent years, the advent of artificial intelligence (AI) and, more specifically, Large Language Models (LLMs) has revolutionized the way we process and analyze textual data. These advancements have opened new avenues for academic research, particularly in the evaluation and enhancement of scholarly communications. This section addresses our third research question (RQ3), which focuses on the application of LLMs to enhance the Openness Score metric of scholarly articles. The Openness Score is a critical measure that assesses the accessibility of research outputs, emphasizing the availability of data, code, and publications to the broader scientific community.

LLMs, with their advanced natural language understanding and generation capabilities, stand out as particularly promising tools for this task that can mimic human-like abilities (Lake et al., 2017). They can interpret the context and content of scholarly articles, identify references to datasets, and evaluate the accessibility of these resources. This ability to process complex linguistic structures and extract meaningful information makes LLMs well-suited for enhancing the Openness Score. By leveraging LLMs, we aim to uncover insights into the open-access status of datasets mentioned within articles, a factor that significantly contributes to the overall openness and reproducibility of research.

Moreover, the application of LLMs in this context is motivated by their potential to automate and scale the analysis of scholarly texts. Traditional methods for assessing the openness of articles often require manual review and are limited by the scale and depth of analysis. LLMs, however, can efficiently process large volumes of text, making it feasible to evaluate a broader array of articles across various disciplines. This capability not only enhances the accuracy of the Openness Score but also contributes to a more comprehensive understanding of research accessibility.

In the following subsections, we will explore the methodologies employed to utilize LLMs for this purpose, including the selection of appropriate models, data collection processes, and analysis techniques. We aim to systematically extract and analyze information related to the accessibility of datasets within scholarly articles, thereby providing an assessment of each article's contribution to the open science movement.

In this section, we address Research Question 3 RQ3, which investigates the application of machine learning (ML) techniques to enhance the Openness Score (OS) by extracting additional information from open-access PDF articles.

Question Answering models

Question Answering (QA) models, pivotal in Natural Language Processing (NLP), are instrumental in interpreting and extracting valuable information from textual data. These models are broadly classified into two categories: Extractive QA and Abstractive QA, each distinguished by its unique approach and underlying architecture.

Extractive QA models aim to provide answers to questions by selecting and extracting a specific span of text from the input document that directly answers the question. These models consist of three components: *Question Processing* analyzes the question and encodes it into a numerical representation, often using transformer-based architectures like BERT or RoBERTa. *Document Processing* reads and encodes the entire document, creating representations for each word or token in the text. *Answer selection* is performed once the document and the question are processed. Extractive QA models rank or score each sentence or phrase in the document based on its relevance to the question. The span with the highest score is selected as the answer.

Abstractive QA models generate answers by summarizing the relevant information in the document and then formulating a concise response in natural language. The architecture of abstractive QA models typically consists of *Question Processing* which encodes the question. *Document Processing* reads the document, but instead of directly extracting text, it generates an abstract representation. *Answer Generation* generates a natural language answer based on the abstract representation and the question. In our research, both model types are explored to determine their efficacy in enhancing the Openness Score of scholarly articles. Extractive QA models offer precision in identifying specific dataset references and their open-access status, while Abstractive QA models provide broader insights into the article's overall openness by generating interpretive summaries. The combined application of these models allows for a comprehensive analysis of textual data, facilitating a more nuanced evaluation of the accessibility and openness of scholarly research.

3.3.1 Data collection & Model Selection

To conduct our experiments, we initiated the process with data collection. In this phase, we aimed to gather a diverse set of open-access articles from the OpenAlex database for our analysis. Our selection criteria encompassed articles from a wide array of academic fields, including Social Science, Humanities, Artificial Intelligence, Law, Natural Science, Physics, Chemistry, Aerospace, Arts, Education, Environmental Science, and Literature. Additionally, we specifically sought articles that referenced datasets, as these are of particular relevance to our research.

The initial step involved the extraction of Digital Object Identifiers (DOIs) for all the selected articles. Subsequently, we employed a script to retrieve these articles from our database, which stored the URLs pointing to their respective PDFs. During this process, we observed that only 38 out of the initially selected articles could be successfully downloaded. This discrepancy can be attributed to the fact that not all articles had corresponding PDF URLs available in our database.

Furthermore, when we utilized the API for downloading, we were able to access and retrieve 43 articles. This difference in the number of downloaded articles is primarily due to variations in how the PDF URLs were structured and categorized within the OpenAlex APIs metadata. This situation echoes the data integrity challenges identified in our investigation of missing information, as previously discussed in response to RQ2.

As a result of these data collection efforts, we proceeded with a dataset comprising 38 articles to conduct our comprehensive study and analysis. To perform our analysis, we have selected the following question-answering models:

- Extractive QA Models
  - Bert

  - Roberta

- Abstractive QA Models
  - GPT series
  - Llama2

This selection of models allows for a comprehensive analysis of the articles, leveraging the strengths of both extractive and abstractive approaches to question answering. By employing a range of LLMs, we aim to capture a detailed picture of how different models can contribute to evaluating and enhancing the Openness Score, thus offering insights into the potential of AI-driven tools in facilitating open science.

3.3.2 Data Analysis

In the data analysis section of our study, we employ a two-step approach to assess the openness score of academic articles. This score reflects the accessibility and usability of the information within each article, particularly focusing on dataset mentions and their availability for public use.

Firstly, we conduct a detailed examination of 38 scholarly articles to establish the ground truth answers for a set of predefined questions. This phase entails an analysis of each article, focusing on extracting accurate and relevant information that directly responds to our queries. These queries are primarily concerned with identifying the presence of datasets within the articles and assessing their accessibility. This step is crucial for evaluating the accuracy and reliability of the responses generated by our question-answering models.

Once these questions are established, next we utilize Large Language Models (LLMs) for question answering (QA). Our approach involves both extractive and abstractive QA models. Extractive QA models are used to locate and retrieve specific segments of text directly from the articles that answer the posed questions. In contrast, abstractive QA models generate summaries or paraphrased content that captures the essence of the information related to the queries.

This dual approach allows for a comprehensive analysis of the text. While extractive models provide direct answers from the content, abstractive models offer a broader interpretation, potentially uncovering more nuanced insights.

Extractive Question Answering

In our analysis, we utilize distinct queries for the extractive to extract relevant information from the articles. The extractive model is tasked with answering the following three questions:

- "What are the names of the datasets used in the article to perform the experiment?"
- "Who are the authors of this article?"
- "Are the datasets used in the experiment in the article open access?"

These questions are designed to gather specific details about the datasets used in the research, the authors of the articles, and the accessibility of the datasets. The precision and direct nature of these questions make them well-suited for an extractive QA model, which excels in locating and retrieving exact information from the text.

To address the set questions, we first employ extractive QA models utilizing HuggingFace's transformer pipelines. These pipelines provide the flexibility to use pre-trained and fine-tuned models, aligning perfectly with our specific QA tasks. We utilize two specific models:

- distilbert-base-cased-distilled-squad (Sanh et al., 2019)
- deepset/roberta-base-squad2 (Deepset, 2022)

We have selected these models because they are not only pre-trained on extensive general language corpora but also fine-tuned on question-answering datasets, making them particularly adept for our QA requirements.

The distilbert-base-cased-distilled-squad model is a distilled version of BERT, offering a balance between computational efficiency and performance. It is further fine-tuned on the Stanford Question Answering Dataset (SQuAD), thus specializing in the task of question answering while retaining the efficiency of DistilBERT.

The deepset/roberta-base-squad2 model leverages the RoBERTa architecture and is fine-tuned on the SQuAD 2.0 dataset. This model is particularly effective in scenarios where answers might not be explicitly stated in the text, using its fine-tuned capabilities to accurately respond to a wider range of questions.

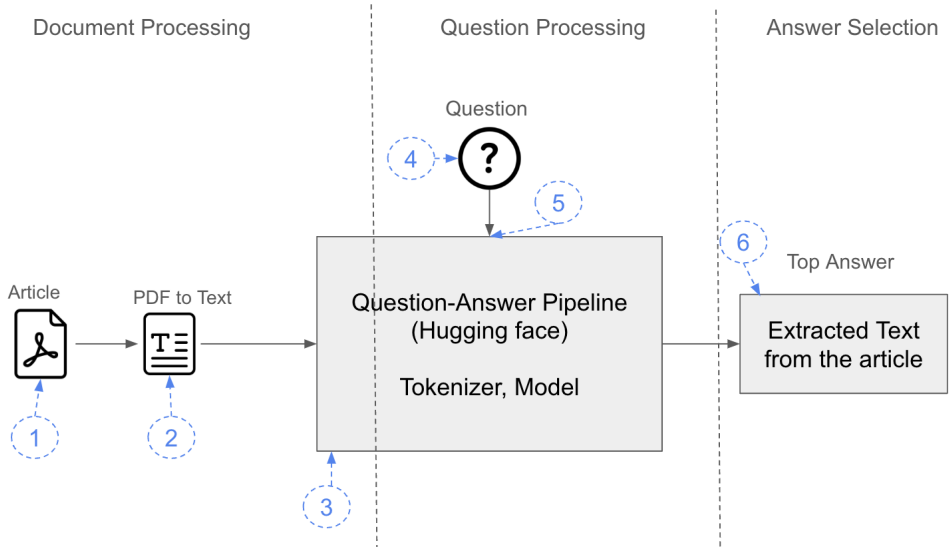

FIGURE 3.4: Extractive Question Answering

The Extractive Question Answering (EQA) process is a multi-step workflow designed to extract precise information from scholarly articles in response to specific questions. This process is divided into three main components: Document Processing, Question Processing, and Answer Selection. Figure 3.4 presents the numbered steps of this workflow, which are detailed as follows:

1. **Document Processing:**

Step 1: The process begins with the retrieval of the target *article*, which is stored in a PDF format. This article contains the information that will be queried by the QA model.

Step 2: The article is in PDF format and is processed using a *PDF to Text* conversion tool, which transforms the article into a text format suitable for the next stages.

Step 3: The converted text is then processed by a tokenizer, which is part of the Question-Answer Pipeline provided by the Hugging Face library.

2. **Question Processing:**

Step 4: The *Question* is encoded into a format that the model can understand, preparing it for the answer selection phase.

Step 5: The pipeline processes both the text of the article and the input question. The tokenizer first breaks down the text into subword tokens, which are then fed into the QA model.

3. **Answer Selection:**

Step 6: The model analyzes the tokens in the context of the question and extracts the segment of the text that best answers the question. This extracted text is what we consider the 'Top Answer' to the input question.

The extracted answers are then evaluated against a ground truth to determine the accuracy and effectiveness of the QA model, and detailed in Section 4.3.1.

Abstractive Question Answering

For the abstractive model, we focus on the query:

- "Are the datasets used in the experiment in the article open access?"

This question, which is similar to one posed to the extractive model, is intended to assess how the abstractive model interprets and paraphrases the text regarding dataset accessibility. The abstractive approach allows for the generation of summaries that might reveal insights not immediately apparent in the direct text, offering a complementary perspective to the extractive analysis.

Retrieval Augmented Generation

Abstractive models can use their learned knowledge to provide expanded answers and are adaptable through prompt engineering to tailor the response format, which is crucial for our openness score calculation. This flexibility allows us to visualize the openness score based on the model outputs.

We compare two models of Large Language Models (LLMs) for this purpose:

- GPT series from OpenAI(commercial)
- LLama2 (Open source)

For the GPT series, we utilize the 'text-davinci-003' and 'GPT 3.5-Turbo' models from OpenAI. These models, although part of a paid service, offer advanced capabilities in language understanding and generation.

In contrast, we chose the LLama2 series for its open-source availability, specifically the 'Llama2 7Billion parameter' version. We chose the 'llama-2-7b-chat.Q4_K_M.gguf' model available on HuggingFace [7] and load onto our server for analysis. The choice of this model is influenced by its resource efficiency, balancing GPU hours, power consumption, and carbon footprint.

[7] https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGUF

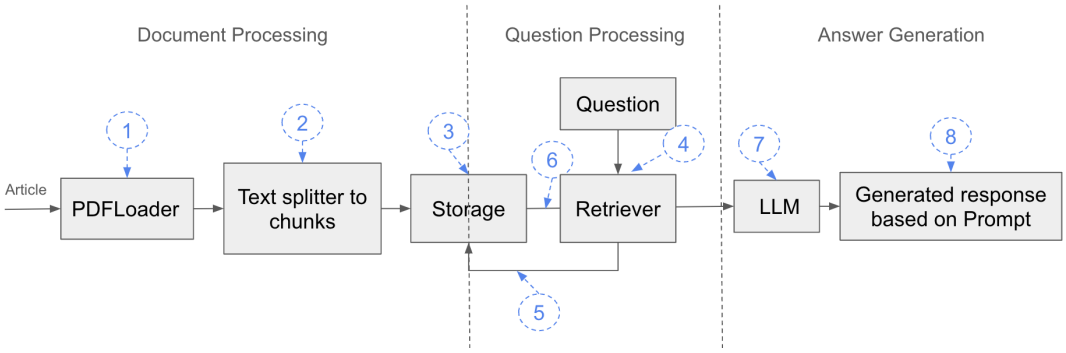

FIGURE 3.5: Abstractive Question Answering with RAG

Due to the token limitations in the open-source LLama2 model, we implement the Retrieval-Augmented Generation (RAG) approach for Abstractive QA. The Retrieval Augmented Generation (RAG) for Abstractive QA employs a mechanism that integrates retrieval of relevant documents from the storage along with language model-based answer generation. The workflow, depicted in Figure 3.5, unfolds as follows:

1. **Document Processing:**

Step 1: The scholarly article, initially in PDF format, is ingested by the *PDFLoader* module which converts it into text format.

Step 2: The resulting text is then segmented into manageable chunks by the *Text splitter*, preparing it for storage and retrieval.

Step 3: The processed text chunks are stored in a repository, making them accessible for the retriever module.

2. **Question Processing:**

Step 4: A question is formulated and encoded, ready to be matched with the most relevant text chunks.

Step 5: The *Retriever* interacts with the storage, utilizing similarity search algorithms to retrieve the 'k' most relevant chunks based on the question's context.

Step 6: The retrieved chunks are then fed into a Large Language Model (LLM).

3. **Answer Generation:**

Step 7: If a prompt is used, the LLM generates a response that aligns with the guidance provided by the prompt.

Step 8: The final output is a generated response that synthesizes information from the selected chunks, articulated based on the prompt (if provided), reflecting a comprehensive understanding of the query.

In our specific implementation, we consider the two most similar chunks as the basis for generating answers. To efficiently manage these chunks, we use 'FAISS' (Facebook AI Similarity Search)(Jegou, Douze, and Johnson, 2017), a library for efficient similarity search and clustering of dense vectors. The chunk size is set at 400 characters, with an overlap of 50 characters to ensure continuity and context preservation in the data. Additionally, we use the newline character ('\n') as a delimiter to separate chunks, aiding in maintaining the structure and readability of the text.

Another crucial parameter set was the 'temperature', which was uniformly maintained at 0.1 across all models. The temperature setting in language models plays a pivotal role in determining the randomness or predictability of the generated responses. A lower temperature value, such as 0.1, steers the model towards more deterministic and possibly more accurate outputs. This setting was chosen to prioritize precision and reliability in the responses, as it reduces the likelihood of generating diverse but potentially less relevant answers. This RAG architecture, combined with our chunking strategy and the use of FAISS for efficient data retrieval, forms the backbone of our approach to leveraging LLMs for extracting meaningful answers from scholarly articles.

In our methodology for abstractive question answering, we extensively utilize the LangChain library (Chase, 2022), a powerful tool for working with Large Language Models (LLMs) and implementing complex NLP tasks. LangChain offers a range of modules that streamline various steps in our data processing and analysis pipeline. The key modules from LangChain that we employ are as follows:

- PyPDFLoader: This module is used for loading and converting PDF documents into text format. It ensures that the scholarly articles in PDF form are accurately transformed into a text format suitable for further processing.
- PromptTemplate: This functionality allows us to create and manage prompt templates for the LLMs. By using templated prompts, we can standardize the queries made to the models, ensuring consistency and reliability in the model responses.
- FAISS: As previously mentioned, we use FAISS for efficient similarity search and clustering of dense vector representations of text chunks. It is crucial for managing the chunk data in our server and enabling rapid retrieval for the RAG model.
- HuggingFaceEmbeddings: This module is employed to generate embeddings for the text data using models from Hugging Face. These embeddings are essential for the functioning of the FAISS vector store and the overall retrieval process.
- RetrievalQA: This component is at the heart of our abstractive QA process. It combines the functionalities of the retriever and answer generator, leveraging the capabilities of LLMs to extract and generate responses based on the retrieved text chunks.
- RecursiveCharacterTextSplitter: We use this tool to split the text into smaller chunks. This splitting is key to managing large text documents and ensuring that the retriever component of the RAG model can efficiently process and select the most relevant sections of text.

By integrating these modules from LangChain, we establish a robust and efficient framework for our abstractive QA methodology. Having established a framework for our abstractive QA methodology with LangChain, we then apply this to conduct our experiment and present our results in Section 4.3.2

Prompt with RAG

Our analytical focus is primarily on answering the third question about dataset accessibility. To achieve precise and categorizable outputs, we employ prompts that

guide the models to respond with a single word: 'green', 'orange', or 'red'. These responses indicate whether the dataset is open to the public, not open, or if the status is ambiguous or unmentioned. For the prompt, the following prompt was provided to the models

```
'You are a helpful, and an expert document analyst.
Use the document as the context {context} to  answer the question
{question}. Only provide single word answers from the context
based on the information about open access status  of the dataset.
If the context mentions that the dataset used is open access
or available for public use, give answer 'Green',
if the dataset used in the context is not open access or limited
reply 'Red', if there is no mention of the status of the dataset
provided or no dataset is used in the article reply 'Orange'.
Your answer should be only one word answer and not have any
text after the answer. If you dont know the answer to a question,
please dont share false information."""
# Answer : single word answer ('green','orange','red').'
```

Additionally, we conduct a comparative analysis between prompted and unprompted responses. This comparison is vital to understanding the impact of prompt engineering on the accuracy and relevance of the information extracted by the LLMs. We aim to determine if the use of prompts leads to more precise results or if it constrains the model's natural language processing capabilities.

Full-text

To evaluate the efficacy of our methodology in diverse scenarios, we selectively deepened our analysis by choosing 15 out of the 38 articles for a more comprehensive examination. Unlike the rest of our study where RAG was employed, for these articles, we provided the full text to the OpenAI models and used the workflow as shown in Figure 3.6. This methodology allows us to assess the OpenAIs' performance in handling complete textual contexts.

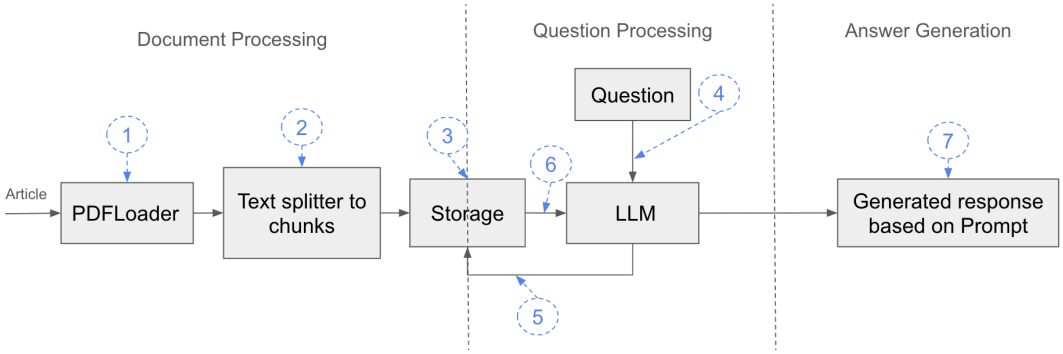

FIGURE 3.6: Abstractive Question Answering with full external document

The abstractive QA model utilizes a workflow that abstracts and interprets information from scholarly articles to generate comprehensive responses. This process, depicted in Figure 3.6, is comprised of the following sequential steps:

1. **Document Processing:**

Step 1: The *PDFLoader* module takes the article in its PDF form and loads it for processing.

Step 2: The *Text Splitter* divides the content into smaller segments, or chunks, optimizing it for efficient processing.

Step 3: These text chunks are then stored in a *Storage* system, from where they can be retrieved as needed during the question-answering process.

2. **Question Processing:**

Step 4: A *Question* is presented to the model, which it encodes in preparation for the answer generation phase.

Step 5: The encoded question is processed by the *LLM (Large Language Model)*, which utilizes the provided context to understand and generate an appropriate response.

3. **Answer Generation:**

Step 6: Utilizing the context and its trained knowledge, the *LLM* synthesizes a response that is reflective of the content's essence.

Step 7: The final output is a *Generated Response* based on the prompt, which captures the model's interpretation and synthesis of the information pertinent to the question.

This approach utilizes the complete text of the document, allowing the model to derive responses from the entire content, rather than being limited to extracting answers from the few chunks as seen in RAG architecture.

We perform analysis on the consistency of the responses provided by OpenAI and LLama2 in response to repeated queries with the same prompt. To accomplish this, each query was rerun for 3 iterations for all 38 articles times under identical conditions, including the same temperature setting(0.1). This approach was designed to test the reproducibility of the model outputs, a critical factor in evaluating the reliability of language models for practical applications. The results of the consistency of the models are provided in Section 4.3.2.

Metrics to evaluate

To evaluate the performance of our models, we compare their outputs against a ground truth, which has been manually annotated from the articles. After conducting our experiments, we meticulously analyze the answers generated by the models and cross-reference them with this ground truth data.

For the evaluation of extractive models, we implement a binary scoring system. For Questions 1 and 2, any model-generated answer that matches, even partially, with the ground truth receives a score of 1. This binary method streamlines the evaluation by focusing on the model's ability to identify correct information. For Question 3, which examines the openness of the dataset used in the article, we assign a score of 0 if the dataset is not open or no dataset is present, a score of 1 if the dataset is open, and a score of -1 for irrelevant answers, for example, when the model extracts a random piece of information from the article. We then determine the True Positives (TP) and True Negatives (TN) and calculate the correct responses for each model. Subsequently, we aggregate these scores to compute an overall success rate for each model, which is detailed in a comparative table in the Results section.

For the abstractive model evaluation, we use accuracy and F1-score as our primary metrics. The ground truth data is scored with binary values: 1 for open datasets and 0 for datasets that are either not open or not mentioned. Correspondingly, the model responses are scored based on their answers; a response indicating an open dataset is scored as 1, while responses indicating a dataset is not open or not found are scored as 0. By aligning these scores with our ground truth, we calculate both the accuracy and F1-score.

Accuracy is the ratio of correctly predicted observations to the total observations. The formula is:

$$Accuracy = (TP + TN)/(TotalObservation) \tag{3.2}$$

The F1 Score is the weighted average of Precision and Recall, particularly useful in cases of imbalanced classes. The formula is:

$$F1\_score = 2 * Precision * Recall/(Precision + Recall) \tag{3.3}$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. It is a measure of the accuracy of the model in classifying a sample as positive. The formula for precision is:

$$Precision = TP/(TP + FP) \tag{3.4}$$

Recall is the ratio of correctly predicted positive observations to all the observations in the actual class. It measures the model's ability to detect positive cases. The formula for recall is:

$$Precision = TP/(TP + FN) \tag{3.5}$$

These calculations also involve the use of a confusion matrix, which is a table used to describe the performance of a classification model. It includes four elements: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These elements are crucial for calculating the above metrics. Our findings from the evaluation of the abstractive QA model are detailed in Section 4.3.2.

In conclusion, this chapter outlines the methodologies and evaluation strategies implemented to address our research questions. It details the rationale behind our data selection, the collection process, and the analysis methods. Furthermore, it explains the choice of metrics used to assess the performance of our models, ensuring a robust and comprehensive evaluation.

Chapter 4

Results

In this chapter, we present the results obtained to address our research questions, which include conducting a comparative analysis of available scholarly APIs, exploring the suitability of API versus an in-house server for implementing the openness score metric application and assessing the potential utility of Large Language Models as an additional source of information in the metric creation process. To address these questions, we will present each research question individually and provide the results from our data analysis.

4.1 RQ1 : Comparative analysis on scholarly APIs

In this section, we present the results of our comparative analysis on four scholarly APIs: Crossref, Doaj, Unpaywall, and Openalex. The objective was to assess the information available in the metadata provided by these APIs and its potential use in calculating an openness score for scholarly articles.

4.1.1 Rate limits

In our investigation of rate limits from the respective API webpages, we uncovered the following information: Crossref offers varying levels of API access based on authentication. Users can access the API anonymously, politely by providing an email address, or with full authentication as part of a paid service. Rate limits differ depending on the level of authentication. To accommodate high-demand services, it is recommended to apply for a Plus account. Doaj suggests that for heavily used services, users consider downloading its data dump as an alternative to circumvent rate restrictions. Unpaywall imposes a limit of 100,000 API calls per day. Additionally, Unpaywall recommends using the publicly available data dump to access data efficiently. Openalex enforces a maximum rate limit of 100,000 calls per day with a cap of 10 requests per second. Premium paid versions of Openalex offer improved rate limits for enhanced service. These rate limits are essential considerations when selecting an API for calculating the openness score. Table 4.1 summarizes the rate limits of the APIs studied.

4.1.2 Access speed

To gauge the access speed for obtaining metadata via the API, we conducted an experiment. We randomly selected 100 scholarly article DOIs, which we then provided to the APIs. This process was repeated for 10 iterations to establish the access speed for retrieving these DOIs. Subsequently, we performed a statistical analysis on the access speed data for these DOIs to discern differences in accessing the APIs for metadata retrieval.

TABLE 4.1: **API Rate Limits.** Summary of rate limits and recommendations for accessing API services.

API	Rate Limit	Recommendations
Crossref	Varies by authentication: Anonymous, Polite (with email), Full (paid)	For high-demand services, apply for a Plus account.
DOAJ	Not specified; heavy use suggests data dump	Consider using the data dump to avoid rate restrictions.
Unpaywall	100,000 calls per day	Use the publicly available data dump for efficient data access.
OpenAlex	100,000 calls per day, 10 requests per second	Premium versions offer improved rate limits for enhanced service.

It is worth noting that, during this experiment, we opted to utilize the polite pool by providing an email address to the API. This choice enhances access speed compared to the anonymous pool, where no identification information is provided to the API. Figures 4.1, 4.2, and 4.3 depict the maximum, minimum, and average access speeds, respectively, for obtaining a DOI from all the APIs.

This analysis of access speed aids in the selection of an API that can deliver data more swiftly, a critical factor, especially for real-time or time-sensitive research. It provides valuable insights into how each API performs in terms of response time.

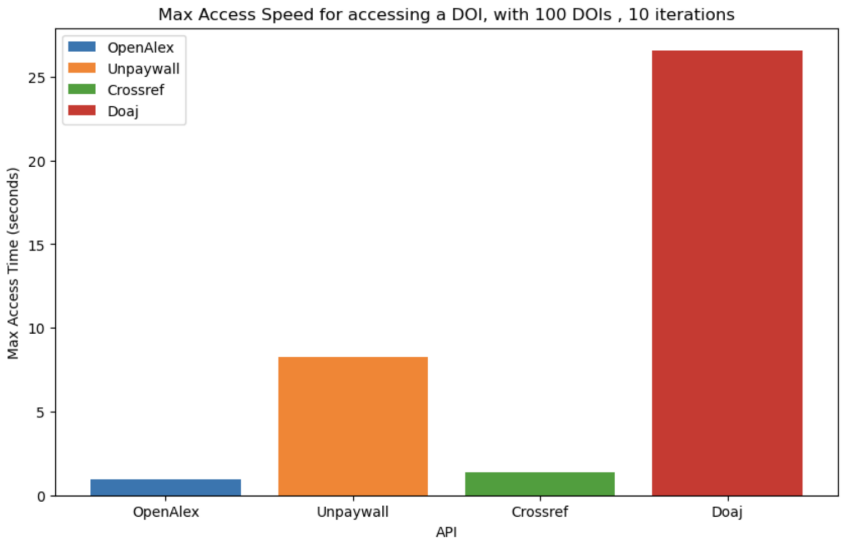

FIGURE 4.1: Max access speed between APIs for 100 DOIs.

4.1.3 Public dump availablility

The availability of public dumps is a crucial factor when considering the accessibility and backup options for API data. In this section we provide an overview of the public dump availability for the selected APIs:

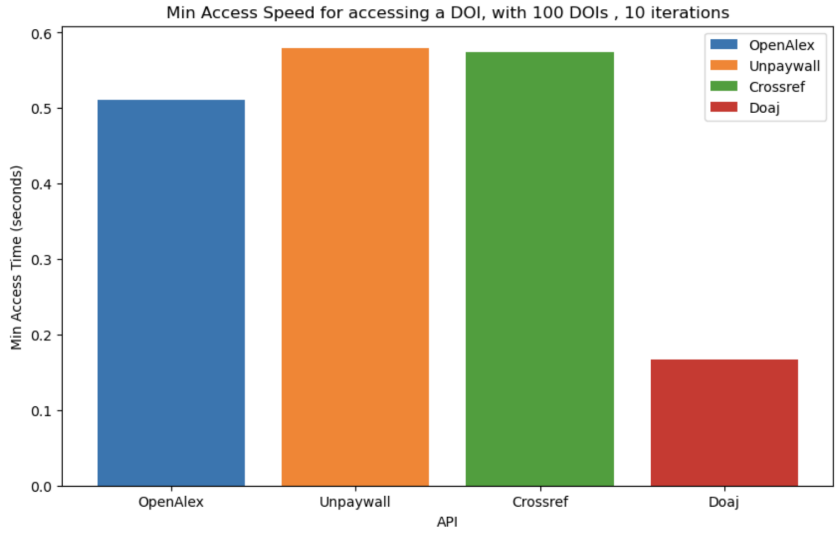

FIGURE 4.2: Minimum access speed between APIs for 100 DOIs.

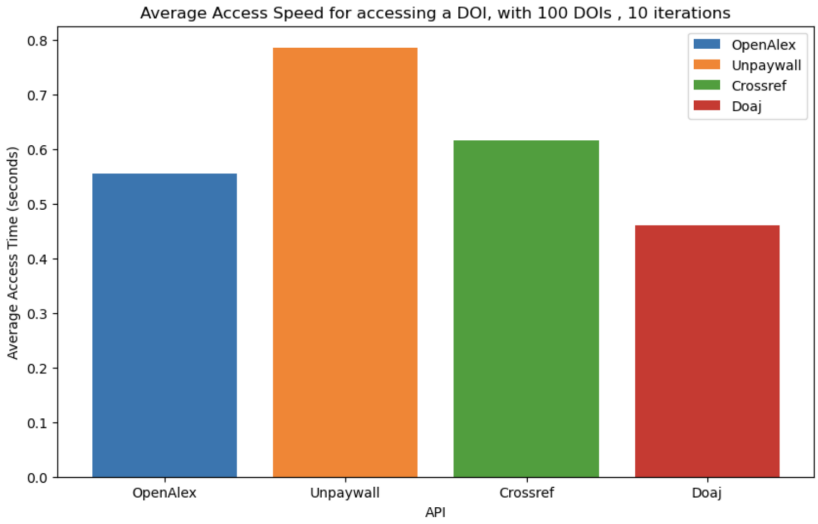

FIGURE 4.3: Average access speed between APIs for 100 DOIs.

Crossref offers the possibility to download their data, which is available as a snapshot hosted on AWS. As of August 2023, the size of the snapshot was 185GB. This snapshot is updated annually and can be downloaded via torrent. Subscribers to the paid version have monthly updates. DOAJ provides a public dump of its data, accessible by requesting it via email and specifying the intended use. Unpaywall has discontinued providing snapshots as their content is now included in the OpenAlex snapshot, which is hosted on AWS and updated monthly. Instead, they offer a zip file containing 350GB of data, which, when unpacked, amounts to 1.7TB of data. Table 4.2, summarizes the findings of the information.

Understanding the availability of public dumps is essential for ensuring data access, backup, and redundancy. These dumps can serve as valuable resources, particularly in mitigating the risk of data loss due to API changes or outages.

TABLE 4.2: **Public Dump Availability.** This table summarizes the availability and characteristics of public data dumps for the APIs studied.

API	Availability	Update Frequency	Size
Crossref	Available as a snapshot on AWS	Annually for free version, Monthly for paid subscribers	185GB (snapshot) -
DOAJ	Available upon request via email	Not specified	6GB
Unpaywall	Included in OpenAlex snapshot	N/A	N/A
OpenAlex	Hosted on AWS, updated monthly	Monthly	350GB (snapshot) -

4.1.4 Data coverage and metadata depth

In this subsection, we delve into the extent of data coverage and metadata depth provided by the selected APIs. Our analysis is designed to answer critical questions about the richness and scope of the data available for scholarly articles through these APIs.

One of the fundamental aspects we examined is the number of records accessible from each API as of 2023. This quantitative metric helps us gauge the breadth of scholarly articles encompassed by each API. Understanding the total count of records is essential in assessing the comprehensiveness of the data sources.

Furthermore, we explored the types of entities or categories of data that each API offers. These entities represent distinct aspects of scholarly articles, such as authors, publications, datasets, and more. Examining these entities is vital in our quest to gather information relevant to the calculation of the openness score metric.

Evaluating the data coverage and metadata depth across these APIs is instrumental in selecting the most suitable sources for our research needs. It enables us to determine which APIs offer the most comprehensive and relevant data for our openness score calculation.

Table 4.3, provides a summary of the information regarding the data coverage and metadata depth obtained from the APIs.

TABLE 4.3: **API Information.** This table provides details about the records, entities, and the number of keys from metadata (DOI) for the APIs studied.

API	Records	Entities	Keys(DOI)
Crossref	145M	7	67
Unpaywall	48M	2	49
Doaj	9M	4	23
OpenAlex	253M	7	201

4.2 RQ2: Comparing efficiency of API and in-house database

To address RQ2, which focuses on whether we should use an API or an in-house database for performing openness score calculations, we conducted a comparison between the two approaches. The primary characteristics under consideration were:

- Response time
- Evaluating data consistency

4.2.1 Response time

In this section, we present an experimental comparison of access speed when retrieving scholarly article information using both the OpenAlex API and an in-house database with the OpenAlex snapshot. Our objective was to assess the performance of these two approaches in terms of speed and efficiency.

For our experiments, we randomly selected 100 articles and conducted 10 iterations. These articles were identified using either their DOI or title. Subsequently, we provided them to both the API and the in-house server, measuring the time it took to complete the entire process for analysis.

Within the API and database, we obtained the article's response and extracted the number of references contained within. Additionally, we retrieved the open access status of the article for visualization purposes. Once we had the reference IDs, we extracted the open access status for each reference. With this information at hand, we proceeded to calculate the openness score, as outlined in Equation 3.1. We generated graphs as shown in Figure 3.2, Figure 3.3 and compiled a table containing all relevant information for visualization. Each iteration was timed, and we collected data for all 100 articles over 10 iterations.

TABLE 4.4: Access Speed comparison between API and Server

Metric	API(sec)	Server (sec)
Average	50.17	0.58
Min	0.421	0.015
Max	540.17	2.47

After collecting the data, we present the average, minimum, and maximum time taken to access an article from both an API and our server in Table 4.4. We can observe that the minimum access time on the server was exceptionally low; this is due to the requested DOI being unavailable in the database, which resulted in an immediate response rather than a delayed retrieval process. We also observed a correlation between the number of references and access speed. To illustrate this relationship, we visualize access speed for selected articles with varying numbers of references. In Figure 4.4, we depict the response time for several articles across different iterations, ranging from no references to 8, 40, and 400 references. Figure 4.5 displays the mean and standard deviation of these response times. Following our analysis of the average speed for calculating the openness score, we create a graph that explores the relationship between the number of references an article has and its average access speed. This analysis aims to identify any correlations between these two variables. In Figure 4.6, we present the observed results.

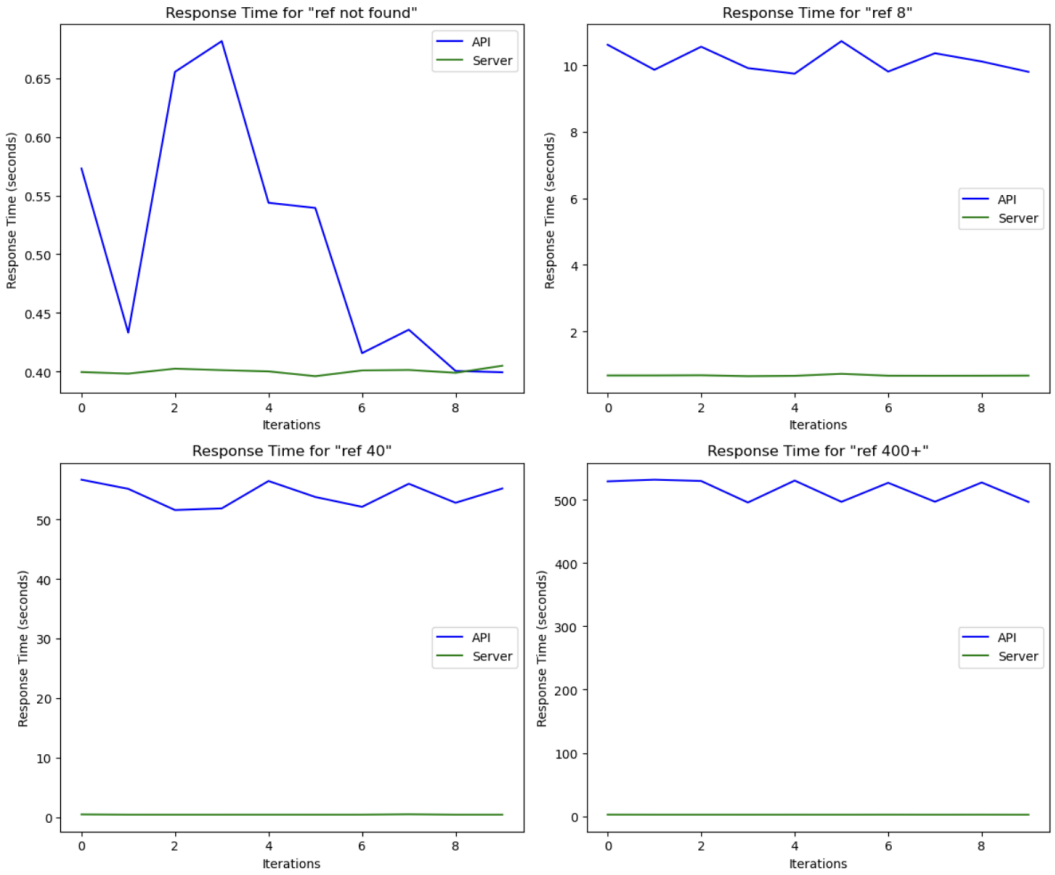

FIGURE 4.4: Response time for 10 iterations for different DOIs with different number of References

4.2.2 Evaluating data consistency

Following our investigation into server and API access speed, we turn our attention to assessing the impact of data inconsistencies caused by outdated information. We aim to quantify the extent of data loss resulting from these inconsistencies.

When comparing data availability between our server and the API, we uncovered notable disparities, as summarized in Table 4.5. The 'Server' column represents the most recent data sourced from OpenAlex as of October 18, 2023, while the comparison is made against API content retrieved on November 14, 2023, with differences highlighted in the 'Diff' column. Additionally, on November 28, 2023, we conducted a follow-up examination of the API to identify discrepancies.

Remarkably, our analysis revealed a discrepancy of 3.5 million records within the API dataset. Subsequently, OpenAlex released an updated snapshot to rectify these irregularities by removing the extra information. In response, we devised and implemented custom scripts to facilitate the comprehensive update of our database, focusing on endpoints other than 'Works.'

For detailed instructions on updating our downloaded snapshot from OpenAlex, we referred to the OpenAlex documentation website, which provides valuable guidance on this process[1].

To update our data, we follow a series of systematic steps. First, we download the updated data from the OpenAlex snapshot, using AWS CLI command with the

[1] https://docs.openalex.org/download-all-data/download-to-your-machine

FIGURE 4.5: Bar chart showing mean and standard deviation of response time depending on the access type

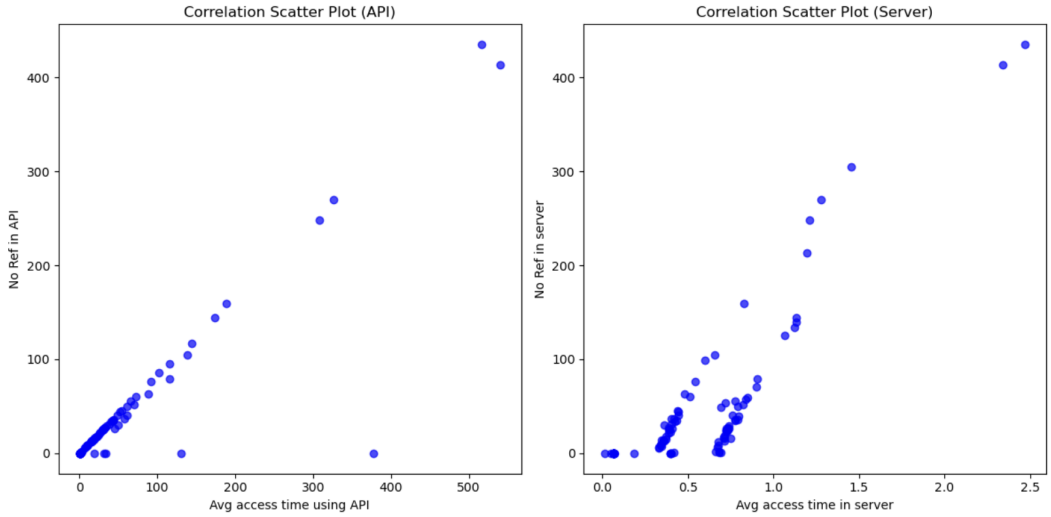

FIGURE 4.6: Scatter plot between the number of references of an article to the average access time

'–delete' option to remove outdated information. Once the data is retrieved, we convert it from JSON format to CSV for further processing. Next, we create a script designed to update our tables efficiently. This script creates a temporary table to house the incoming updates, which are then integrated into the actual table while looking for conflicts. To maintain data integrity, we proceed by identifying and deleting the merged IDs. To accomplish this, we utilize another script specifically crafted to gather all the merged IDs from the 'merged_id' file. We systematically collect all the relevant IDs for each endpoint, excluding 'works,' and subsequently deleting these entries from the respective tables.

During the process of removing these entries from our database, we took the opportunity to conduct an analysis comparing the efficiency of individual row deletion and subsequent commitment to the database versus batch deletion and commitment. We have included the results of this experiment in our Appendix chapter for reference in Section 7.1.

After updating our database, we present the results in Table 4.6. In this table, O_Ser represents the old database, and U_Ser represents the updated database. The "O_Ser-U_Ser" column highlights the differences between the two after updating the

TABLE 4.5: **API Information.** This table provides server and API records for different endpoints

Endpoints	Server	API (Nov 14)	Diff	API (Nov 28)	(API-API)
Works	245207435	246139651	932219	246537492	397841
Author	93003987	89468168	-3535819	89565600	97432
Institutions	106956	107247	291	107252	5
Concepts	65073	65073	0	65073	0
Publishers	10250	10250	0	10250	0
Sources	247955	248650	695	248643	-7

server. Additionally, the "API-U_Ser" column illustrates the differences observed following the update process.

TABLE 4.6: **Updated database** This table presents the updated values in our database.

Endpoints	O_Ser	U_Ser	API Nov28	O_Ser-U_Ser	API-U_Ser
Works	245207435	245207435	246537492	0	1330057
Author	93003987	89516053	89565600	3487934	49547
Institutions	106956	107246	107252	-290	6
Concepts	65073	65073	65073	0	0
Publishers	10250	10250	10250	0	0
Sources	247955	248643	248643	-688	0

For Research Question 2, which involved a comparative analysis between the API and our in-house database, we have identified important insights. Our results reveal that the server's access speed outperforms the API, indicating that accessing data directly from our server is the quicker option. However, it is noteworthy that the API consistently maintains a higher level of data currency, making it a valuable resource for up-to-date information. These findings provide a clear understanding of the trade-offs between speed and data freshness when choosing between our in-house database and the API. Further exploration of the implications and considerations associated with these results will be presented in Chapter 5.2.

4.3 RQ3: Evaluation of LLMs on scholarly articles

In this section, we present our findings from the evaluation of Large Language Models (LLMs) on a curated dataset of 38 scholarly articles. Our analysis focuses on the models' ability to accurately extract and interpret information relevant to the Openness Score of these articles. We utilized human annotation to establish a ground truth for comparison, assessing each model's performance against this benchmark.

4.3.1 Extractive QA model

Evaluation

Using the workflow in Figure 3.4, we extract the response from the extractive models for the 3 questions.

- Question1: "What are the names of the dataset used in the article to perform the experiment?"

- Question2: "Who are the authors of this article?"
- Question3: "Are the datasets used in the experiment in the article open access?"

In evaluating the responses from the extractive models, we compare them to the established ground truth, assigning scores based on accuracy. A score of 1 is given for responses that are either fully correct or partially correct, while a score of 0 is assigned to incorrect responses. For instance, in response to Question 2, which inquires about the authors of the article, a response that identifies at least one author correctly is awarded a 1. Similarly, for Question 3, which asks about the open-access status, a response indicating 'open-access'—without specifying whether it refers to the dataset or the article—is deemed sufficiently accurate to merit a score of 1.

In Table 4.7, we present the number of correct answers that each model provides and calculate the Success rate for each question. We observe that for Question 1, distilbert-base-cased-distilled-squad(DistilBert) performs better, whereas, for Questions 2 and 3, deepset/roberta-base-squad2(RoBERTa performs better. The bar graph in Figure 4.7 visually shows these findings, offering a clear comparison of the models' success rates across the three questions. The graph not only aids in understanding the quantitative differences in model performance but also serves as a basis for deeper qualitative analysis.

TABLE 4.7: Comparison of DistilBert and RoBERTa for 38 articles, including success rates

	Correct Answers		**Success Rate (%)**	
	DistilBert	**RoBERTa**	**DistilBert**	**RoBERTa**
Question 1	11	8	28.95	21.05
Question 2	10	15	26.32	39.47
Question 3	10	15	28.95	39.47

Furthermore, our analysis extended to the models' efficiency, measured by the time taken to respond to queries. DistilBert showed a notable advantage in speed, potentially offering a more efficient solution for real-time analysis of scholarly articles. The specifics of these findings are detailed in Table 4.8, indicating DistilBert's faster response times.

Model	**Average Time (s)**	**Max Time (s)**	**Min Time (s)**
DistilBert	21.83	131.96	6.53
RoBERTa	42.43	262.09	12.41

TABLE 4.8: Comparison of Time Taken by DistilBert and RoBERTa Models

4.3.2 Abstractive QA model

RAG Evaluation

In this section, we present the results obtained from employing the GPT series and Llama models, as outlined in the methodology described in Section 3.3.2.

The LLama2 model, being open-source, has limitations regarding the amount of text it can process. For the OpenAI Language Learning Models (LLMs), we employed the Text-embedding-ada-002-v2 model for embedding. We also compared

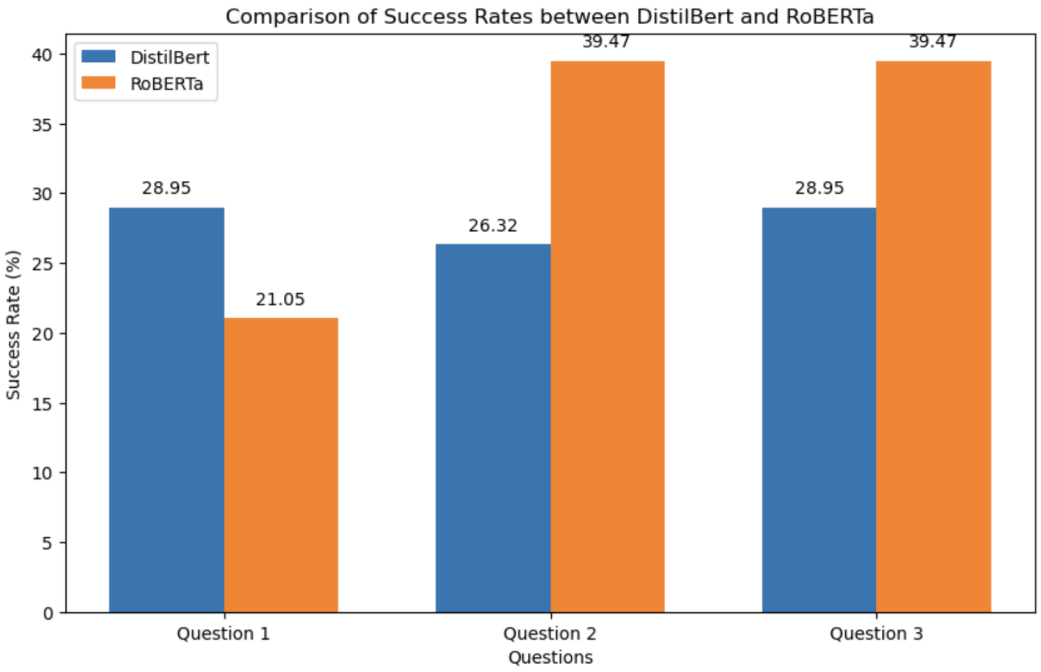

FIGURE 4.7: Success rates of DistilBert & RoBERTa for the three questions

the performance of their older model, text-davinci:003, with the newer and more efficient GPT 3.5-Turbo, which is cost-effective relative to other models. Following the Retrieval-Augmented Generation (RAG) architecture as depicted in Figure 3.5, all models utilized the FAISS similarity search. This component embeds text chunks and stores them on our server. Upon receiving a query, in this case, Question 3, the retriever selects two documents most relevant to the query. These documents, along with the model's internal knowledge, are used to generate a response.

TABLE 4.9: Correct Response for Models on Question 3 (RAG) with 38 articles

Model	**True Positives**	**True Negatives**
llama-2-7b-chat.Q4_K_M.gguf	18	4
Text-davinci:003	16	15
GPT 3.5-Turbo	18	14

From Table 4.9 we observe the following, Llama2-7B achieved a total of 22 correct responses (18 TP + 4 TN), indicating a solid capacity to identify relevant answers even in the absence of explicit contextual guidance. Text-davinci:003 presented a more balanced performance with 31 correct responses (16 TP + 15 TN). This model's higher count of True Negatives, in particular, suggests a strong ability to discern irrelevant information without the aid of prompts, possibly indicating superior comprehension or filtering mechanisms. Whereas GPT 3.5-Turbo recorded 32 correct responses (18 TP + 14 TN), showing a comparable proficiency to Text-davinci:003 in processing and responding to the question. The performance suggests that GPT 3.5-Turbo can effectively navigate the question's context and make informed decisions about relevance and irrelevance on its own.

Prompt Evaluation

This subsection presents results obtained by integrating prompts into our RAG architecture.

The prompt instructed the LLM to classify the data access as Green (open access), Orange (undetermined status), or Red (closed access). When we have datasets with a Creative Commons license we expect them to be classified as Green, permitting acquisition with author permission, and a score of 1 is provided. If a response of 'Orange' is provided by the model for the query, indicating uncertainty, a score of zero is assigned. The 'Orange' category was used for ambiguous cases where the information provided did not conclusively indicate the access status of a dataset. Future work could refine the prompt design to address this scenario and also provide a different scoring mechanism to handle the unknown category.

TABLE 4.10: Comparison of Correct Responses with Prompt for Different Models

Model	True Positives	True Negatives
llama-2-7b-chat.Q4_K_M.gguf(Llama2-7B)	21	9
Text-davinci:003	23	11
GPT 3.5-Turbo	7	15

Table 4.10, presents the results of introducing prompts in our models. We observed Llama2-7B with prompts, and produced 30 correct responses (21 TP + 9 TN), demonstrating an improved ability to identify relevant answers, likely due to the additional context clarifying the question's intent. Text-davinci:003 excelled in this scenario, achieving the highest total of 34 correct responses (23 TP + 11 TN) among the models. This suggests that Text-davinci:003 not only benefits from prompts in identifying relevant information but also maintains a strong capability to filter out irrelevant details, indicating a highly effective use of contextual cues. However, GPT 3.5-Turbo showed a mixed reaction to prompts, with a total of 22 correct responses (7 TP + 15 TN). While the number of True Negatives is commendable, the significant drop in True Positives suggests a potential overemphasis on avoiding false positives at the expense of missing relevant information, highlighting a cautious or conservative approach when prompts are present.

Full text evaluation

In our investigation into the performance of various Large Language Models (LLMs) on scholarly articles, we extended our analysis to compare how these models perform when provided with the entire text of an article, as opposed to just specific sections retrieved via queries. This examination aimed to uncover potential disparities in model effectiveness when dealing with comprehensive versus segmented textual data. The analysis focused on a subset of 15 articles, with particular attention given to models' ability to accurately determine dataset usage within these texts.

Table 4.11 presents the outcomes of this comparison, showcasing the number of correct responses generated by each model under different conditions: analyzing the full text, using the Retrieval-Augmented Generation (RAG) architecture without prompts, and employing RAG with prompts. Notably, the models examined include OpenAI's Text-davinci:003 and GPT 3.5-Turbo, along with a consideration of the open-source Llama2-7B model, and for completeness, the performance of DistilBert and RoBERTa is also included for the 15 articles.

TABLE 4.11: Comparison of Responses for 15 articles

Model	Full Text	RAG (No Prompt)	RAG (Prompt)
Text-davinci:003	12	11	14
GPT 3.5-Turbo	6	12	6
llama-2-7b-chat.Q4_K_M.gguf	-	9	11
DistilBert	3	-	-
RoBERTa	7	-	-

The results indicate a varied performance across models, with Text-davinci:003 showing a notable resilience and adaptability to the full-text analysis, contrasting with the performance dips observed in GPT 3.5-Turbo under similar conditions.

Consistency & Efficiency of Responses

Our findings revealed a noteworthy pattern: both the Llama2 and OpenAI models provided identical responses across all three iterations for each of the 38 articles. This consistent behavior was observed despite the variety of content in the articles. The temperature setting of 0.1 likely played a significant role in this outcome, as it typically leads to more deterministic outputs by reducing the likelihood of generating diverse responses. This consistent result across multiple reruns highlights the predictability and stability of these models when operating with a low-temperature setting.

In the comparative analysis of response times between the OpenAI GPT-3.5-Turbo, Text-davinci:003, and Llama2-7B models, a significant disparity was observed. The data, as presented in Table 4.12, shows the response time for each model in seconds.

TABLE 4.12: Response Times of LLM on RAG with Prompt

Model	Max Time (s)	Min Time (s)	Avg Time (s)
GPT 3.5-Turbo	6.21	0.75	1.26
Text-davinci:003	11.61	0.52	0.96
Llama2-7B	77.53	37.03	53.54

The results indicate that the Text-davinvi:003, has the quickest response time with 0.96 seconds on average, whereas GPT 3.5-Turbo model has a response time of 1.26 seconds which are faster, compared to the Llama2-7B model, which averages 53.55 seconds per response.

When it comes to cost analysis, the OpenAI models exhibit varying price, with GPT 3.5-Turbo and Text-davinci:003 charging $0.32 and $5.19 respectively for the experiments conducted. However, LLama2 emerges as a distinctly advantageous option in terms of cost-effectiveness. Given its open-source status, LLama2 incurs no direct cost for usage, making it a preferable choice.

LLama2's open-source attribute significantly reduces the financial barriers often encountered in advanced research and aligns perfectly with the principles of open science. Its free accessibility ensures that researchers and institutions around the world can leverage this advanced tool without the hindrance of cost. This not only underscores LLama2's role in cost-saving but also highlights its contribution to a more open, equitable, and collaborative research landscape. The elimination of usage fees democratizes access to cutting-edge language processing capabilities,

thus reinforcing LLama2's position as a valuable asset in the research community's toolkit.

TABLE 4.13: Cost Comparison of Language Models

Model	Cost (USD)
GPT 3.5-Turbo	0.32
Text-davinci:003	5.19
Embedding model	0.21
Llama2	0.00

Accuracy & F1 scores

In the subsequent phase of our analysis, we quantitatively evaluated the performance of the selected LLMs using F1 score, accuracy, precision, and recall. These metrics provided a comprehensive understanding of each model's effectiveness in accurately identifying and interpreting information within scholarly articles. The evaluation focused on comparing the models' performance across different scenarios, including full-text analysis and the use of the Retrieval-Augmented Generation (RAG) architecture, both with and without custom prompts.

Table 4.14, compares the accuracy, precision, recall, and F1 score for the Llama2-7B model, Text-davinci:003, GPT 3.5-Turbo for the 38 articles.

TABLE 4.14: Evaluation Metrics for Text-davinci:003, GPT 3.5-Turbo, and Llama2-7B Models(38 articles)

	Text-davinci:003		GPT 3.5-Turbo		Llama2-7B	
	No Prompt	Prompt	No prompt	Prompt	No prompt	Prompt
Accuracy	82%	89%	84%	58%	58%	79%
Precision	100%	85%	95%	100%	62%	78%
Recall	70%	100%	78%	30%	78%	91%
F1 Score	82%	92%	86%	47%	69%	84%

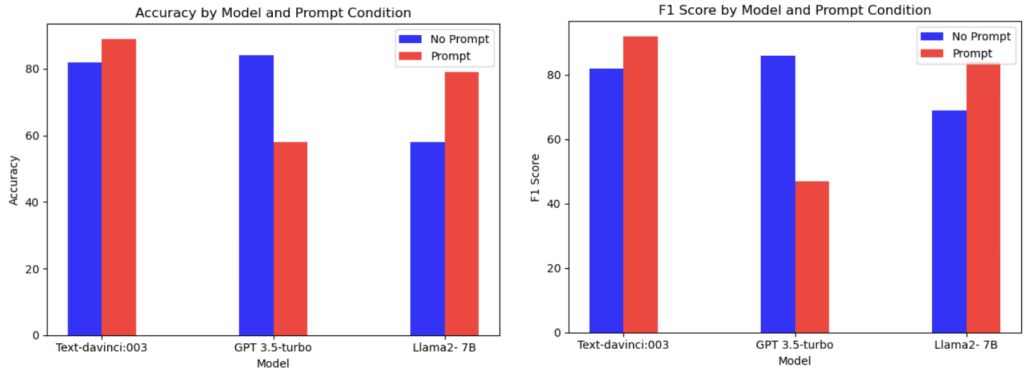

FIGURE 4.8: F1 score and Accuracy of models with and without prompt

In our analysis of the impact of custom prompts on model performance, we observed a distinct pattern in the F1 scores and accuracy rates, as illustrated in the bar

graphs in Figure 4.8. For the Text-davinci:003 model, the introduction of prompts resulted in a notable increase in both accuracy (from 82% to 89%) and F1 score (from 82% to 92%). Whereas, GPT 3.5-turbo did not benefit from prompts with accuracy decreasing from 84% to 58% and F1 score seeing a decrease from 86 to 47%. Interestingly, the Llama2-7B model showed the most significant improvement with prompts, with the F1 score jumping from 69% to 84% and accuracy increasing from 58% to 79%. These findings are crucial as they highlight the effectiveness of well-designed prompts in enhancing model performance.

Next, we focused on evaluating the GPT 3.5-turbo and Text-davinci:003 models using full-text articles. Here, we considered 15 articles from the 38 articles. Then we compare the various variations (prompt, no prompt, and full text) of these models. These conditions were designed to test the models' adaptability and accuracy in different analytical scenarios, offering insights into their practical utility for scholarly communication. Table 4.15 presents the obtained metrics, including accuracy, precision, recall, and F1 score, across 15 articles. These metrics collectively offer a nuanced view of each model's performance, shedding light on their strengths and areas for improvement in processing complex academic texts. Figure 4.9 visualizes these metrics, highlighting the comparative performance of the models across the tested conditions, thereby facilitating a visual comprehension of their effectiveness in different analytical contexts.

TABLE 4.15: Performance Metrics for GPT 3.5-Turbo and Text Davinci Models(15 articles)

	GPT 3.5-Turbo			**Text-davinci:003**		
	No Prompt	**Prompt**	**Full-Text**	**No prompt**	**Prompt**	**Full-Text**
Accuracy	80.00%	40.00%	40.00%	73.33%	93.33%	80.00%
Precision	100.00%	100.00%	100.00%	100.00%	91.67%	100.00%
Recall	72.73%	18.18%	18.18%	63.64%	100.00%	72.73%
F1 Score	84.21%	30.77%	30.77%	77.78%	95.65%	84.21%

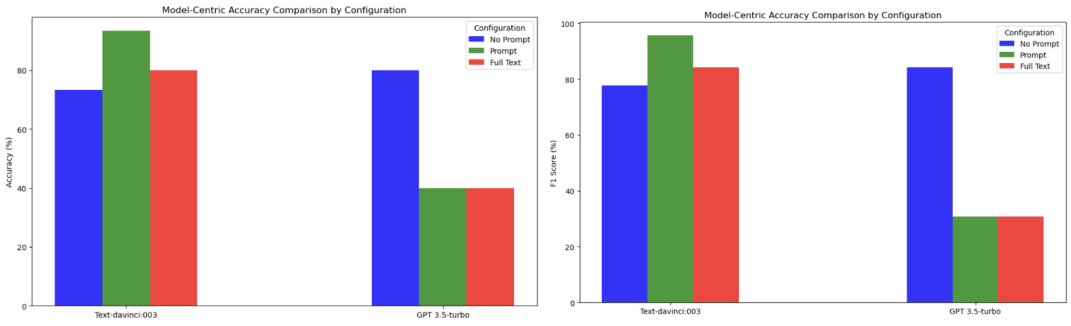

FIGURE 4.9: F1 score and Accuracy of models with, without prompt, and Full-Text Comparison

Building upon these insights, this chapter consolidates the results of our extensive experimental investigations. To address our first research question (RQ1), we conducted a comparative analysis of the Crossref, Unpaywall, Doaj, and OpenAlex APIs, focusing on key characteristics such as rate limits, access speed, and metadata depth, which are integral to the computation of an openness score for scholarly articles. For our second research question (RQ2), we evaluated the efficiency of using the OpenAlex API against a relational database populated with an OpenAlex

data snapshot. This evaluation included an analysis of the speed of openness score calculation and an assessment of data integrity between the API and our database solution. Lastly, in response to our third research question (RQ3), we compared the performance of extractive and abstractive question-answering models to determine their suitability and effectiveness as additional resources in the calculation of the openness score metric.

Chapter 5

Discussion

In this chapter, we delve into the results of our experiments designed to address our three research questions. We analyze the data gathered through our methodology, providing insights for each research query.

5.1 RQ1: Comparative analysis on scholarly APIs

Selecting the right API is crucial for an API-dependent application. Our results underscore the importance of rate limits and data availability when making this decision. As per the suggestion of the APIs, having the possibility to download the data in case of failures is key for us. Since Unpaywall snapshots are available with Openalex, we decided to download the snapshots from Openalex as well as Doaj to compare the results of the API and the downloaded data.

After requesting data from DOAJ, we closely examined its content. Although it contained all the basic metadata, our expectations of additional, more detailed information were not met. The minimalistic nature of DOAJ's data made it unsuitable for our openness score calculation. Consequently, we decided to focus our future analysis on Openalex as the primary API of choice.

Analyzing the metadata depth of each API revealed the varying amount of information available. DOAJ and Crossref APIs offer article abstracts as one of their key features. Openalex, Crossref, and Unpaywall provide vital information about the open-access status of articles, a critical factor for constructing the openness metric. Both Openalex and Crossref metadata include references to the main article. The metadata from Unpaywall and Crossref APIs is also present in Openalex. Additionally, Openalex provides URL links to open-access PDFs, a valuable resource for our analysis in response to RQ3.

When interpreting the results of API access speed, we conducted 10 iterations of fetching metadata for 100 DOIs. The plots clearly indicate that DOAJ consistently exhibits lower average access speed compared to other APIs. This discrepancy can be attributed to the notably smaller size of DOAJ's responses. Furthermore, examining the maximum access time, DOAJ had a maximum response time of 26 seconds to access a DOI, signifying that it is not a reliable option for fast data retrieval. In contrast, Openalex consistently performed well in terms of access speed throughout our analysis.

Table 5.1 offers a concise overview of the comparative analysis of scholarly APIs, illustrating OpenAlex's superior record count and the shared rate limit advantage with Unpaywall. It emphasizes the variation in data accessibility, download options, and the size of datasets across APIs, thereby OpenAlex exhibits strong potential as a suitable candidate for further investigation in the context of our second research question analysis.

TABLE 5.1: Comparison of Scholarly APIs

	OpenAlex	Unpaywall	Crossref	DOAJ
Records	253M	48M	145M	9M
Avg Access Speed(sec)	0.55	0.79	0.6	0.45
Rate limit	100,000 calls/ day	100,000 calls /day	Varies by authentication	Not specified
Download	Snapshot AWS	OpenAlex snapshot	Torrent-update once a year	Yes
Size	350GB	-	185GB (2023)	6GB

5.2 RQ2:Comparing efficiency of API and in-house database

In this section, we discuss and interpret the results from our experiment conducted to answer Research Question 2 RQ2.

Upon analyzing **response times** in our experiment, we examined the average, maximum, and minimum response times for accessing 100 articles over 10 iterations in Table 4.4. Notably, the response times for APIs were considerably higher. To delve into the reasons behind this discrepancy, we explored the relationship between the number of references and response time. Figure 4.6 illustrates a clear correlation: as references increase, so does response time. While a similar trend was observed with the server, the increase in response time was less pronounced. These results indicate that when using APIs, the latency is influenced by the number of references associated with an article.

During our experiment involving 100 articles, we made a significant observation regarding database query optimization. Initially, our application allowed users to provide any part of the title, which was implemented using a SQL query with wildcard matching (LIKE operator). However, this approach incurred longer response times. To investigate the reasons behind this, we conducted an experiment, the results of which can be found in Chapter 7.2.

In Table 4.5, we illustrate the **disparities between the server and API** datasets. With the server containing a recent snapshot (October 18, 2023), it included 932,219 new works that were not present in the API. Furthermore, when comparing the server with the API two weeks later, an additional 397,841 new works had been added to the API. These observations highlight that the API contains more accurate and up-to-date information compared to the server. During this analysis, we also noted that the snapshot from October 2023 showed an excess of 3.5 million records in the server. However, with the release of the subsequent snapshot on November 24, 2023, these extra rows were removed, as documented in their release notes [1].

Table 4.6 presents the outcomes of updating the OpenAlex tables in our server, excluding the works tables. In the "O_Ser-U_Ser" column, which represents the difference between the old server and the updated server, we successfully updated our server by removing the 3.5 million redundant records. Nevertheless, after the update, we observed that there were still 49,547 new authors added when comparing the server's data as of November 24, 2023. The difference in data between the API

[1] https://github.com/ourresearch/openalex-guts/blob/main/files-for-datadumps/standard-format/RELEASE_NOTES.txt

and our server is striking. This strongly suggests that relying on the API is a favorable choice due to its superior data consistency and up-to-date information.

When considering only latency, hosting our server proves to be the more efficient choice. However, maintaining timely updates within our PostgreSQL database poses a challenge. Furthermore, to sustain fast response times, the deployment of distributed servers may be necessary, which can incur higher management costs. Decisions regarding these aspects must align with the needs of the Openness Score team.

In light of the comprehensive results obtained from our experiments, it becomes evident that a deeper exploration of alternative database systems, such as graph databases or NoSQL databases, is warranted. As discussed in Li and Manoharan (2013), a comparative study between different databases can be performed to choose the right one. While our research focused on the performance of an in-house database and an API, the choice of the right database technology is a critical decision for building any application. Each database type offers unique advantages and challenges, and by considering graph databases or NoSQL databases, we can gain a more comprehensive understanding of how different database architectures may better suit the requirements of our application. Such an investigation could shed further light on optimizing data retrieval, storage, and maintenance processes to enhance the overall efficiency and effectiveness of our system.

Additional findings from our experiments reveal that when attempting to retrieve information from both the API and server by providing the DOI or title, the API returns no article, even when the article is present in the database. This unexpected behavior stems from the fact that some article titles contain characters such as ',' or ';', which are not accepted by the OpenAlex API. Consequently, when utilizing the API for calculating the openness score, it is imperative to implement exception handling to address these situations.

In pursuit of building a robust and accurate solution for calculating the Openness Score, we propose a workflow as shown in Figure 5.1. The process involves checking an article's validity through its DOI or title in the in-house database. If neither is unavailable, we verify its validity through the API. If the article remains inaccessible, we halt the calculation. Otherwise, we proceed to calculate the Openness Score using the API.

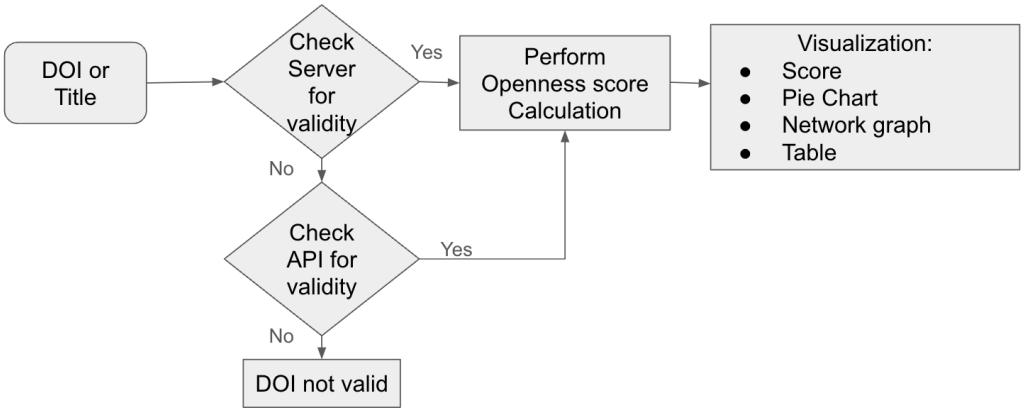

FIGURE 5.1: Architecture to handle missing data in in-house database compared to API

For Research Question 2, which involved a comparative analysis between the

API and our in-house database, we have identified important insights. Our results reveal that the server's access speed outperforms the API, indicating that accessing data directly from our server is the quicker option. However, it is noteworthy that the API consistently maintains a higher level of data currency, making it a valuable resource for up-to-date information.

These findings provide a clear understanding of the trade-offs between speed and data freshness when choosing between our in-house database and the API.

5.3 RQ3: Evaluation of LLMs on scholarly articles

In this subsection, we discuss the outcomes of our evaluation concerning the LLMs employed to address our RQ3.

Table 4.7 offers a comparative analysis between the "distilbert-base-cased-distilled-squad"(DistilBert) and "deepset/roberta-base-squad2"(RoBERTA) models, across three specific queries. The performance is visually depicted in Figure 4.7. It is observed that RoBERTA outperforms DistilBert in responding to Question 2 and Question 3, while the latter takes the lead on Question 1. The strength of extractive models lies in their proficiency in sourcing direct responses from text, particularly when questions are tightly aligned with the text's wording. However, they encounter challenges when the text is formatted differently or if the question is not directly present in the article. In such cases, these models often extract answers from various parts of the article and choose the text segment with the highest score as the answer. In Table 5.2, we present a selection of responses obtained from the models for question 3. Upon examination, it is evident that in the fourth article, the response generated by the model is not relevant to the question. In contrast, the other three articles yielded more relevant and accurate responses. Overall, RoBERTA boasts an overall success rate of 33.33%, whereas DistilBert lags slightly at 27.19%.

TABLE 5.2: Q3 response from Extractive QA models

Article	**DistilBert**	**RoBERTa**
10.1093_molbev_msab120.pdf	provided the original work is properly cited	provided the original work is properly cited"
10.1371_journal.pcbi.1004219.pdf	open access article distributed under the terms of the Creative Commons Attribution License	open access
10.1371_journal.pone.0231924.pdf	Creative Commons Attribution License	provided the original author and source are credited
10.1186_s13054-019-2347-3.pdf	Recommendations are grouped behind the rationale for key decision points	Equivalent data for prasugrel appear not to have been published

The observed performance differences can be attributed to factors such as the model architecture, training data, and the specific nature of the questions. RoBERTA's more complex model structure and extensive training with SQUAD.2.0 may contribute to its higher accuracy, especially in understanding the context or nuances of certain types of questions. DistilBert, in contrast, may not reach the same level

of accuracy as RoBERTA, but it stands out for its faster processing speed as shown in Table 4.8. This difference in speed is attributed to DistilBert's design, which is optimized for efficiency and reduced computational load, making it a more viable option in scenarios where quick response times are critical. This trade-off between accuracy and speed is a crucial consideration in our study, reflecting the need to balance these two factors based on the specific requirements of the task at hand. Thus, while RoBERTA may be more effective in terms of accuracy, DistilBert's advantage in speed cannot be overlooked, especially in applications where time efficiency is a priority.

Although extractive models are proficient in obtaining information directly from texts, they are limited in their ability to generate responses that are specifically tailored to unique requirements. A notable feature of these models, however, is their provision of a confidence score with each response. This score is a crucial metric that can be utilized to assess the reliability of the responses. For instance, a lower score may indicate a lower level of reliability or relevance of the response to the query. In our current experiments, this aspect of utilizing the confidence score to gauge response validity was not explored. However, it presents a promising avenue for future research. Incorporating this score-based assessment method could significantly enhance the utility of extractive models by enabling a more nuanced evaluation of the responses they generate.

To be able to control the response we used abstractive models.

Table 4.9 presents the correct responses for three models – Text-davinci:003, GPT 3.5-Turbo, and llama-2-7b-chat.Q4_K_M.gguf(Llama2-7B) – when querying without prompts on a set of 38 articles.

Text-davinci:003 model achieved 31 correct responses out of 38, indicating a high level of effectiveness in extracting relevant information without additional prompting. The performance suggests that this model is adept at understanding and responding to queries in a context where explicit guidance (in the form of prompts) is not provided. With 32 correct responses, GPT 3.5-Turbo slightly outperformed Text-davinci:003. This could be attributed to its advanced model architecture and training, which might be more attuned to extracting and interpreting information from scholarly articles, even in the absence of explicit prompting. Llama2-7B model scored 22 correct responses, which is noticeably lower compared to the GPT models. This difference could stem from various factors such as the model's training data, its inherent architecture, or its specific strengths and weaknesses in processing the type of content present in the articles.

In Table 4.10, the performance of the same models is evaluated, but this time with the integration of prompts in the queries.

Text-davinci:003 with Prompt's performance slightly increased from 31 to 34 correct responses when prompts were introduced. This indicates that the model's natural language understanding capabilities might be sufficiently robust that the addition of prompts does enhance its performance. Whereas in GPT 3.5-Turbo with Prompt, there was a notable decrease in correct responses from 32 to 22. This significant drop might imply that the prompts used could have constrained the model's natural response generation or introduced a bias that led to less accurate answers.

Llama2-7B with Prompt's performance increased significantly from 22 to 30 suggesting that this model's output is affected by the addition of prompts. This could reflect a different approach to query processing or a different balance between training on structured versus open-ended queries.

In our analysis, we observed a notable decrease in the correctness of responses from the GPT 3.5 -Turbo model when prompts were introduced. A key factor contributing to this decline appears to be the nature and structure of the prompts themselves. Specifically, the prompts did not include comprehensive examples or clear guidance on handling certain types of data, such as datasets under Creative Commons licenses.

Our prompts were structured with the expectation that a response indicating an 'open access' dataset would be labeled as 'Green'. However, in cases where the model's response was 'Orange' or 'Unknown', which did not align with our expectation, these responses were assigned a negative score. This scoring approach adversely impacted the overall performance metrics of the models.

The shortfall in performance with prompts can be attributed to the lack of explicit examples in the prompts that could guide the model in recognizing and responding accurately to queries about datasets with specific licensing statuses. Without such examples, the models were less equipped to handle queries where the licensing status of the data was a key consideration. As a result, the models often defaulted to responses that were deemed less accurate according to our scoring criteria.

This finding underscores the importance of carefully crafting prompts, especially when dealing with nuanced or specific information requirements. The prompts must be designed not only to elicit the desired type of response but also to provide clear examples or guidelines that align with the model's processing capabilities and training. In instances where the model encounters data scenarios that are not explicitly covered in its training or in the provided prompts, its ability to generate accurate responses can be significantly compromised.

Therefore, enhancing the effectiveness of prompts in future experiments involves incorporating more detailed and representative examples, particularly for cases that require a nuanced understanding of dataset types and licensing statuses. By doing so, we can better align the model's responses with our scoring system and improve the overall accuracy of the results.

In Table 4.15, the results from the inclusion of full text instead of using RAG are studied and indicate notable differences in the performance of GPT 3.5-Turbo and Text-davinci:003 across different configurations. Text-davinci:003 consistently outperforms GPT 3.5-Turbo in scenarios involving prompts and full-text inputs, showcasing superior adaptability and effectiveness in understanding and responding to varying contexts. The stark difference in recall and F1 Scores between the two models, particularly in the Prompt and Full-Text configurations, suggests that Text-davinci:003 is more adept at comprehensively capturing and addressing the nuances of the input data.

The Prompt configuration stands out as a significant factor influencing performance, with Text-davinci:003 demonstrating its prowess in leveraging additional context to enhance answer relevance and completeness. In contrast, GPT 3.5-Turbo's performance suggests potential limitations in effectively utilizing prompts and full-text inputs to guide its answer-generation process.

When we performed our analysis on the consistency of the response of OpenAI and Llama2 models it is important to note that the temperature setting of the models was set to 0.1. This low-temperature value may have contributed to the higher precision observed in the responses from the various models. This setting likely influenced the models' tendency to provide more deterministic answers, which is a key aspect to consider in interpreting the results. To check if there are any hallucinations or random answers from the model, we looked at running the command for 3 iterations for the 38 articles, we observed that all the answers were identical, this

could be the result of using temperature= 0.1, which avoids providing any hallucination.

The graphs depicting F1 scores and accuracy in Figure 4.8 provide a visual affirmation of the benefit of incorporating prompts into the models' querying process. Particularly, the Text-davinci:003 and Llama2-7B models showed improved performance with the use of prompts, underscoring the potential of prompts to guide models toward more accurate outputs. The discrepancy in the GPT 3.5-turbo model's performance suggests that the influence of prompts may vary depending on the model's architecture and the nature of the task. The increase in F1 scores indicates that prompts can significantly enhance precision and recall, leading to more reliable results. However, the observed decrease in the GPT 3.5-turbo model's F1 score upon prompting calls for a deeper examination of the interaction between model design, task complexity, and the prompting mechanism. These insights pave the way for further research into tailored prompt engineering to optimize the performance of language models across various applications.

The radar chart in Figure 5.2 provides a visual summary of the comparative performance of the Text-davinci:003, GPT 3.5-Turbo, and Llama2-7B models across different evaluation metrics: Accuracy, F1 Score, Precision, and Recall. The chart indicates that the Text-davinci:003 model with prompting (Text-davinci:003 P) has a strong lead in most metrics, showcasing its robustness in understanding and generating responses. The GPT 3.5-Turbo model without prompting (GPT 3.5-Turbo NP) also performs well, particularly in terms of precision.

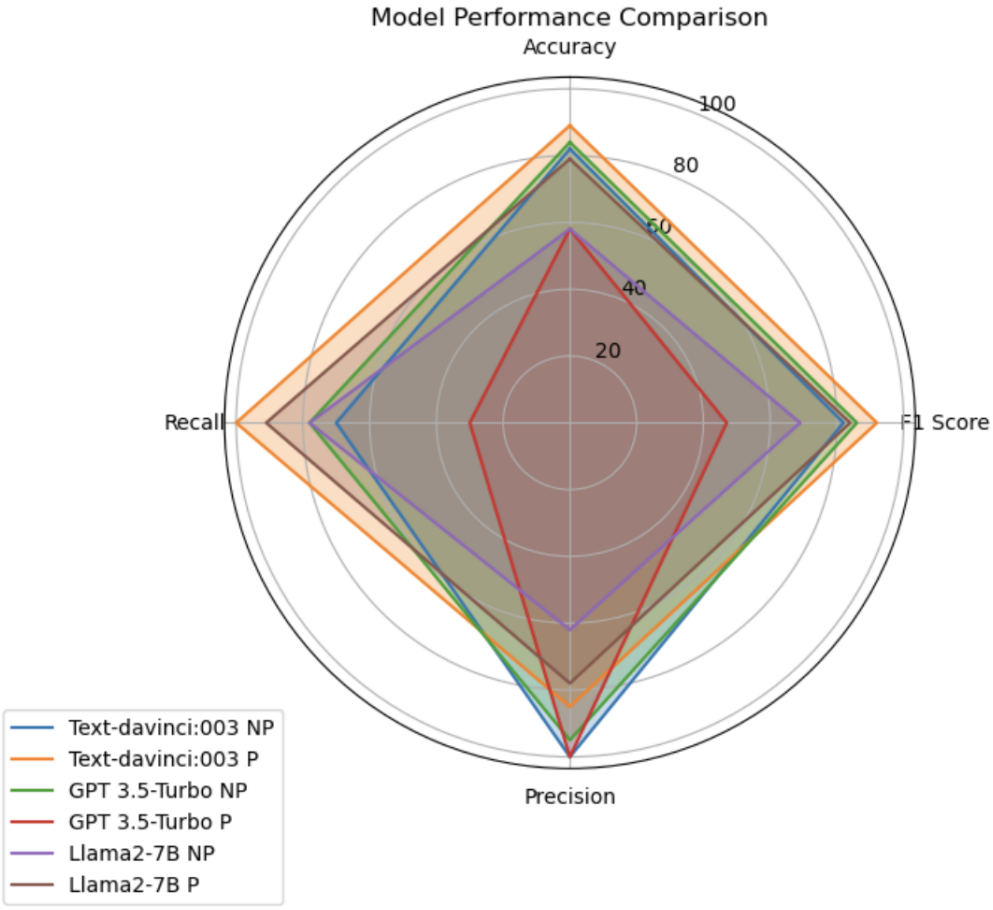

FIGURE 5.2: Radar Chart to compare different LLM

Despite the Llama2-7B model not leading in most metrics, its performance with prompting (Llama2-7B P) is noteworthy, especially considering its cost-effectiveness. Since the Llama2-7B model is open-sourced, it provides a significant advantage in terms of accessibility and budget, making it a practical choice for our research framework. Furthermore, the flexibility and adaptability of the Llama2-7B model through prompt engineering show its potential utility in a wide range of applications.

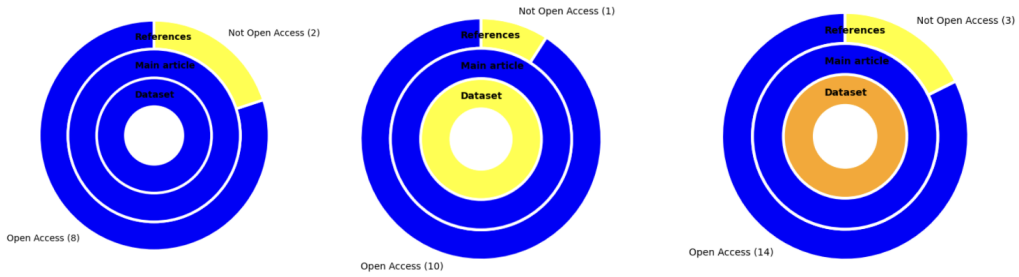

FIGURE 5.3: Openness score visualization with LLM enhancement

Given these considerations, the Llama2-7B model with prompting is selected for our framework, aligning with open science and ensuring cost-efficient, yet effective, performance in large-scale language processing tasks. We follow the framework proposed in Figure 5.4. For the visualization of openness, which also incorporates answers from the LLM, we propose using a nested pie chart. We present a simulation of this graph. When the response from the LLM is 'open,' indicating the dataset is open, we color the inner pie 'blue.' Conversely, when the LLM responds with 'closed,' we assign the inner pie color 'yellow,' and 'orange' when an 'unknown' response is obtained from the model. Simulations of this visualization are shown in Figure 5.3.

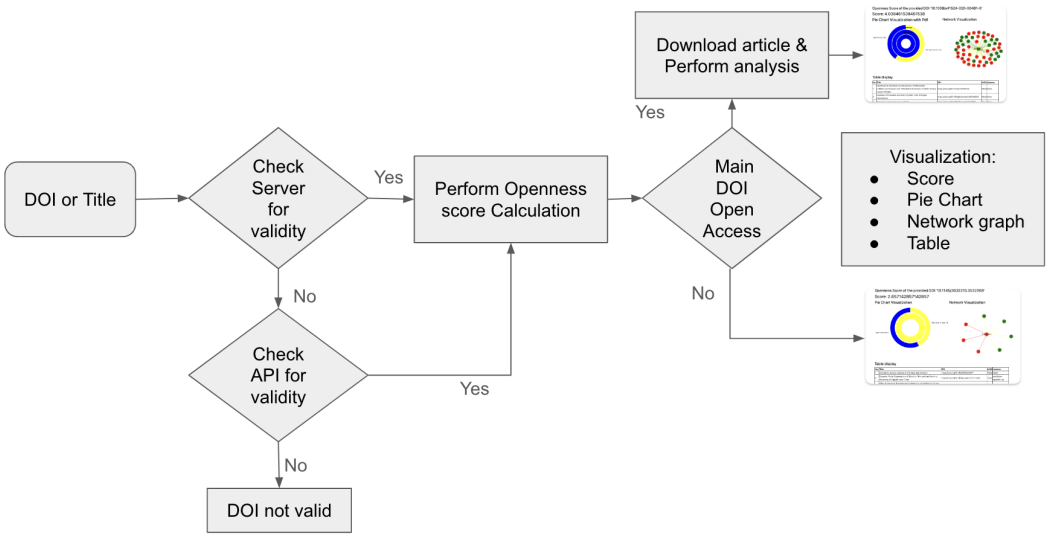

FIGURE 5.4: Framework with LLM enhancement

In conclusion, this chapter has provided a comprehensive analysis of the experiments conducted to address three pivotal research questions concerning the assessment of openness in scholarly publications. Through a detailed examination of

scholarly APIs, the efficiency of API endpoints versus locally hosted servers, and the effectiveness of Large Language Models (LLMs) in extracting valuable information, this research offers valuable insights into the landscape of scholarly communication and its openness.

Chapter 6

Conclusion

6.1 Conclusion

In this thesis, we conducted a comprehensive analysis addressing three research questions aimed at developing a framework for calculating the openness score of scholarly articles. Our motivation stemmed from a project undertaken by the Openness Score team, partners of Swiss universities, who aspire to promote openness in scholarly research and embrace the FAIR principles within the academic community. These metrics serve to encourage researchers to adopt more transparent practices in their studies. To achieve this goal, we embarked on our analysis with three research questions, which form the basis for our openness score metric.

In Research Question 1 (RQ1), we examined several scholarly APIs providing metadata on articles. Our analysis encompassed various characteristics, including access speed, rate limits, data download options, and metadata content, all essential for our openness score calculation. Among these APIs, Openalex emerged as the top performer, showcasing promising potential for our metric.

Turning our attention to Research Question 2 (RQ2), we downloaded the Openalex snapshot content into an in-house server within a PostgreSQL database. Here, we conducted experiments to understand the differences between using an API and an in-house server. Our findings highlighted a crucial trade-off: the API offers accurate and up-to-date updates, while the PostgreSQL database allows faster information access. As a result, we proposed a workflow (as depicted in Figure 3.1) that first checks the server for article presence and, if unavailable, resorts to the API to ensure comprehensive data for openness score calculation. This approach balances data accessibility with speed.

In Research Question 3 (RQ3), we delved into the possibility of incorporating PDF text content to enhance the openness score calculation. Leveraging Natural Language Processing (NLP), we explored question-answering techniques on PDF texts to gather valuable information. Our analysis included extractive models such as DistilBert and Roberta from Hugging Face, as well as abstraction models like the GPT series and Llama2. While extractive models showed promise, we observed that question framing played a crucial role in their performance. Consequently, we explored abstractive models, particularly ChatGPT and Llama2, an open-sourced Large Language Model (LLM) trained on 7 billion parameters, capable of generating answers based on context. Despite Llama2's performance being slightly behind the GPT series, we propose adopting a framework that incorporates the Llama2 model for openness score calculation (as illustrated in Figure 5.4).

Furthermore, this research has played a pivotal role in shedding light on the distinctions between utilizing APIs and servers for accessing scholarly article data. It

has underscored the importance of balancing data accuracy and up-to-date information with access speed, offering a valuable framework for future studies in this domain.

While this thesis has made significant strides in exploring the potential of Large Language Models (LLMs) for openness score calculations, it is crucial to acknowledge that more experiments and investigations are needed. Specifically, conducting experiments with a broader range of prompts and questions will be essential in harnessing the full potential of LLMs for this purpose. These efforts can further refine our understanding of how LLMs can be effectively integrated into the openness score metric, ultimately contributing to the advancement of transparency and openness in scholarly research.

In conclusion, this research not only provides practical solutions and insights for openness score calculations but also sets the stage for further exploration in this evolving field. By bridging the gap between API and server performance and delving into the capabilities of LLMs, we aim to foster a culture of openness and accessibility within the scholarly community, promoting a more transparent and collaborative research environment.

6.2 Limitation & Future Work

To answer our Research Question 1 (RQ1), we focused on specific APIs, such as Crossref, Unpaywall, Doaj, and OpenAlex, due to time constraints and practical considerations. However, it is important to acknowledge that there are other APIs available, including PubMed and arXiv, that could provide additional information and enable further comparisons of their characteristics.

To answer our Research Question 2 (RQ2), we focused on comparing the efficiency of using an in-house PostgreSQL database and the openalexAPI for accessing research article data. However, it is important to acknowledge the limitations of this study.

One limitation lies in the limited exploration of different database types. This study primarily compared the performance of PostgreSQL and the openalexAPI. While it yielded valuable insights, future studies could broaden their scope by including a more diverse range of database types, such as graph databases or NoSQL databases. Exploring these alternatives could provide a more comprehensive understanding of their performance in the context of openness score calculations.

Additionally, our study primarily centered on response time as a critical performance metric. While response time is undoubtedly crucial, it is essential to recognize that our analysis did not encompass a comprehensive evaluation of throughput. Evaluating throughput could offer further insights into system capacity and performance under varying workloads, enriching the depth of our research findings.

Another limitation worth noting is related to the works table in our PostgreSQL database. While we diligently updated other tables to reflect the latest data snapshots from openalex, the works table remained unchanged during the experimental process. Future research should consider updating all relevant tables to ensure data consistency across the entire database.

While conducting our experiment for research question 3 (RQ3), offers valuable insights, but it is important to acknowledge its limitations. We experimented using a relatively small dataset of only 38 articles, all containing mentions of datasets. This limited dataset size could affect the generalizability of our findings. Additionally,

the dataset was labeled and extracted by a single person, introducing the possibility of human error in identifying the correct answers.

Furthermore, our study evaluated models based on a specific set of three questions. Expanding the range of questions could provide a more comprehensive understanding of model performance. Additionally, there is a wide variety of language models available, and our analysis only covered a subset of them. Future work could involve testing a broader selection of models to gain a more extensive view of their capabilities.

In terms of metrics, we provided binary answers, but exploring more nuanced scoring mechanisms, such as partial scores for answers, could offer a more refined evaluation of model performance. Moreover, the articles used in the experiment were not preprocessed before being provided to the models. While we attempted to use regular expressions to remove reference sections, this method proved imperfect. Future research could explore more advanced techniques for preprocessing, including the removal of stop words and the extraction of specific sections from PDF articles, to enhance the quality of input data.

In conclusion, while our study provides valuable insights, it has limitations including dataset size, human error, and metric choice. Future work should address these limitations, expand dataset size, diversify data sources, explore more APIs or models, and refine performance metrics. Preprocessing techniques should also be investigated to enhance experiment accuracy and efficiency, allowing for further advancements in our research.

Bibliography

Allam, Ali Mohamed Nabil and Mohamed Hassan Haggag (2012). "The question answering systems: A survey". In: *International Journal of Research and Reviews in Information Sciences (IJRRIS)* 2.3.

Arvidsson, Simon and Johan Axell (2023). "Prompt engineering guidelines for LLMs in Requirements Engineering". In.

Baron, Joseph and Sanjay Kotecha (2013). "Storage options in the aws cloud". In: *Amazon Web Services, Washington DC, Tech. Rep*.

Besançon, Lonni et al. (2021). "Open science saves lives: lessons from the COVID-19 pandemic". In: *BMC Medical Research Methodology* 21.1, pp. 1–18.

Burgelman, Jean-Claude et al. (2019). "Open science, open data, and open scholarship: European policies to make science fit for the twenty-first century". In: *Frontiers in big data* 2, p. 43.

Chakravorty, Nishant et al. (2022). "Open science: Challenges, possible solutions and the way forward". In: *Proceedings of the Indian National Science Academy* 88.3, pp. 456–471.

Chase, Harrison (2022). *LangChain*. https://github.com/langchain-ai/langchain. Accessed: 26 January 2024.

Chaudhuri, Surajit (1998). "An overview of query optimization in relational systems". In: *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pp. 34–43.

Chen, Jiawei et al. (2023). "Benchmarking large language models in retrieval augmented generation". In: *arXiv preprint arXiv:2309.01431*.

Deepset (2022). *RoBERTa-based SQuAD 2.0 Model*. https://huggingface.co/deepset/roberta-base-squad2. Hugging Face Model Hub.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Dhakal, Kerry (2019). "Unpaywall". In: *Journal of the Medical Library Association: JMLA* 107.2, p. 286.

Gillioz, Anthony et al. (2020). "Overview of the Transformer-based Models for NLP Tasks". In: *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, pp. 179–183.

Hendricks, Ginny et al. (2020). "Crossref: The sustainable source of community-owned scholarly metadata". In: *Quantitative Science Studies* 1.1, pp. 414–427.

Jacobsen, Annika et al. (2020). *FAIR principles: interpretations and implementation considerations*.

Jegou, H, M Douze, and J Johnson (2017). "Faiss: A library for efficient similarity search". In: *Library provided by Facebook Research*.

Jiao, Chenyue, Kai Li, and Zhichao Fang (2023). "How are exclusively data journals indexed in major scholarly databases? An examination of the Web of Science, Scopus, Dimensions, and OpenAlex". In: *arXiv preprint arXiv:2307.09704*.

Jung, Min-Gyue et al. (2015). "A study on data input and output performance comparison of mongodb and postgresql in the big data environment". In: *2015 8th international conference on database theory and application (DTA)*. IEEE, pp. 14–17.

Kidwell, Mallory C et al. (2016). "Badges to acknowledge open practices: A simple, low-cost, effective method for increasing transparency". In: *PLoS biology* 14.5, e1002456.

Lake, Brenden M et al. (2017). "Building machines that learn and think like people". In: *Behavioral and brain sciences* 40, e253.

Lewis, Patrick et al. (2020). "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in Neural Information Processing Systems* 33, pp. 9459–9474.

Li, Yishan and Sathiamoorthy Manoharan (2013). "A performance comparison of SQL and NoSQL databases". In: *2013 IEEE Pacific Rim conference on communications, computers and signal processing (PACRIM)*. IEEE, pp. 15–19.

Liu, Yinhan et al. (2019). "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692*.

McKiernan, Erin C et al. (2016). "How open science helps researchers succeed". In: *elife* 5, e16800.

Mervin, R (2013). "An overview of question answering system". In: *International Journal Of Research In Advance Technology In Engineering (IJRATE)* 1, pp. 11–14.

Mons, Barend et al. (2017). "Cloudy, increasingly FAIR; revisiting the FAIR Data guiding principles for the European Open Science Cloud". In: *Information services & use* 37.1, pp. 49–56.

Morrison, Heather (2017). "Directory of open access journals (DOAJ)". In: *The Charleston Advisor* 18.3, pp. 25–28.

Pentz, Ed (2001). "CrossRef: a collaborative linking network". In: *Issues in science and technology librarianship* 10, F4CR5RBK.

Priem, Jason, Heather Piwowar, and Richard Orr (2022). "OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts". In: *arXiv preprint arXiv:2205.01833*.

Rizzetto, Elia and Silvio Peroni (2023). "Mapping bibliographic metadata collections: the case of OpenCitations Meta and OpenAlex". In: *arXiv preprint arXiv:2312.16523*.

Roumeliotis, Konstantinos I and Nikolaos D Tselikas (2023). "ChatGPT and Open-AI Models: A Preliminary Review". In: *Future Internet* 15.6, p. 192.

Sanh, Victor et al. (2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *NeurIPS EMC^{2}Workshop*.

Sciences, Engineering National Academies of, Medicine, et al. (2018). "Open science by design: Realizing a vision for 21st century research". In.

Stonebraker, Michael and Greg Kemnitz (1991). "The POSTGRES next generation database management system". In: *Communications of the ACM* 34.10, pp. 78–92.

Touvron, Hugo et al. (2023). "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288*.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Visser, Martijn, Nees Jan Van Eck, and Ludo Waltman (2021). "Large-scale comparison of bibliographic data sources: Scopus, Web of Science, Dimensions, Crossref, and Microsoft Academic". In: *Quantitative science studies* 2.1, pp. 20–41.

Wang, Kuansan et al. (2020). "Microsoft academic graph: When experts are not enough". In: *Quantitative Science Studies* 1.1, pp. 396–413.

Wilkinson, Mark D et al. (2016). "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific data* 3.1, pp. 1–9.

Ye, Junjie et al. (2023). "A comprehensive capability analysis of gpt-3 and gpt-3.5 series models". In: *arXiv preprint arXiv:2303.10420*.

Zhang, Yue et al. (2023). "Siren's song in the ai ocean: A survey on hallucination in large language models". In: *arXiv preprint arXiv:2309.01219*.

Chapter 7

Appendix

7.1 Efficiency Comparison of Row Deletion in PostgreSQL

In this appendix, we present the detailed findings and analysis of our experiment comparing two methods of deleting rows from a PostgreSQL table and committing the changes. The goal of this experiment is to determine which method is more efficient in terms of time taken for the deletion and committing operations. We conducted a series of iterations for each method and collected data for analysis.

We performed the following two methods of row deletion and data collection:

Method 1: Delete One Row at a Time and Commit Immediately for 1000 rows

- For each iteration, we deleted one row from the PostgreSQL table and immediately committed the change for 1000 rows.
- We recorded the time taken for 10 iteration.

Method 2: Delete 1000 Rows in Batch and Commit Once

- For each iteration, we deleted 1000 rows from the PostgreSQL table and committed the changes.
- We recorded the time taken for 10 iteration.

We calculated the mean and standard deviation for both methods to analyze the efficiency of row deletion and committing.

In Table 7.1, we show the results of our experiment. The results indicate that Method 2, which involves deleting 1000 rows in batch and committing once, is significantly more efficient in terms of time taken. The lower mean and standard deviation values for Method 2 as shown in Table 7.2 suggest that it provides more consistent and faster performance compared to Method 1.

This analysis demonstrates that batch processing and deleting multiple rows from the PostgreSQL table is more efficient. The primary reason for the improved efficiency is that batch processing allows multiple rows to be deleted simultaneously, reducing the overhead of individual disk write operations required for each deletion, which results in shorter execution times.

In conclusion, our experiment demonstrates that deleting rows in batches and committing once (Method 2) is a more efficient approach in PostgreSQL compared to deleting rows one by one and committing immediately (Method 1). This finding is based on the lower mean and standard deviation values for Method 2, indicating faster and more consistent performance.

TABLE 7.1: **Comparison of Deletion and Commit Times.** This table shows the time taken for deletion and commit operations for two different methods: deleting one row at a time and deleting 1000 rows in batch before committing.

Iteration	Delete 1 Commit (seconds)	Delete 1000 Commit (seconds)
1	13.460	0.953
2	13.978	0.820
3	13.977	0.990
4	14.963	1.006
5	15.769	1.343
6	15.395	1.237
7	13.837	1.629
8	14.794	1.772
9	14.963	1.573
10	15.492	1.709

TABLE 7.2: Mean and Standard Deviation for Row Deletion and Committing Methods

Method	Mean (seconds)	Standard Deviation (seconds)
Method 1	14.50	1.30
Method 2	0.76	0.33

7.2 Query Optimization

In this section, we compare the query plans and access times for two SQL queries that involve searching for rows in a database table with an indexed w.title column. The queries are as follows:

Query 1: Using LIKE with a Wildcard Pattern

```
SELECT
    referenced_work.doi AS DOI,
    referenced_work.title AS TITLE,
    oa.is_oa AS IsOA,
    hv.license AS License
FROM
    openalex.works AS w
JOIN
    openalex.works_referenced_works AS
    rw ON rw.work_id = w.id
JOIN
    openalex.works AS referenced_work
    ON rw.referenced_work_id = referenced_work.id
JOIN
    openalex.works_open_access oa
    ON oa.work_id = referenced_work.id
JOIN
    openalex.works_primary_locations hv
    ON hv.work_id = referenced_work.id
WHERE w.title LIKE '%Buddhists, Shamans,
```

```
and Soviets: Rituals of History in
Post-Soviet Buryatia%';
```

Query 2: Using = Operator

```
SELECT

        referenced_work.doi AS DOI,
        referenced_work.title AS TITLE,
        oa.is_oa AS IsOA,
        hv.license AS License
    FROM
        openalex.works AS w
    JOIN
        openalex.works_referenced_works AS
        rw ON rw.work_id = w.id
    JOIN
        openalex.works AS referenced_work
        ON rw.referenced_work_id = referenced_work.id
    JOIN
        openalex.works_open_access oa
        ON oa.work_id = referenced_work.id
    JOIN
        openalex.works_primary_locations hv ON
        hv.work_id = referenced_work.id

        WHERE w.title = 'Buddhists, Shamans,
        and Soviets: Rituals of History in
        Post-Soviet Buryatia';
```

Access Time Comparison:

We perform access time comparison when we use both methods and present our findings. We chose 10 random articles and found the access time for performing openness score calculation using both methods the access time with operator = and the exact pattern matching using LIKE with wildcards'%'.

TABLE 7.3: Server Times for 'Like' & '='

Article	'Like' Time	'=' Time
1	13508.999171	1.070869
2	12891.091120	0.876983
3	12936.179617	0.930006
4	12791.037560	0.914162
5	14301.807437	0.969082
6	14310.659077	0.606237
7	14296.754336	0.633421
8	14417.923326	0.882877
9	14315.915732	1.195073
10	14364.505460	1.028949

Table 7.3 shows that the access time is high for columns in big tables even when the column is indexed. The experiment with these queries reveals that Query 2 (using =) has significantly faster access times compared to Query 1 (using LIKE with

wildcards). This performance difference arises due to the nature of pattern-matching operations required by LIKE, which can be resource-intensive, particularly when applied to large datasets.

Query Execution Plans:

To compare the reason for higher access time, we use Explain Query in postgres to obtain the query plans as shown in Figures 7.1 and 7.2. For Query 1, which utilizes the LIKE clause with a wildcard pattern, the database query optimizer (Chaudhuri, 1998) chooses a more complex execution plan. As seen, they perform entire table scan instead of using the index. The increased complexity and broader scope of the search in this plan are reflected in its higher estimated cost and larger row count.

```
                                                                   QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=25938393.17..70301896.16 rows=232084 width=134)
   Workers Planned: 2
   ->  Nested Loop  (cost=25937393.17..70277687.76 rows=96702 width=134)
         ->  Nested Loop  (cost=25937392.59..69450587.72 rows=96702 width=221)
               ->  Nested Loop  (cost=25937391.90..68612628.11 rows=96702 width=188)
                     ->  Parallel Hash Join  (cost=25937391.20..67774668.38 rows=96702 width=32)
                           Hash Cond: (rw.work_id = w.id)
                           ->  Parallel Seq Scan on works_referenced_works rw  (cost=0.00..39235753.44 rows=991056644 width=64)
                           ->  Parallel Hash  (cost=25937266.35..25937266.35 rows=9988 width=32)
                                 ->  Parallel Seq Scan on works w  (cost=0.00..25937266.35 rows=9988 width=32)
                                       Filter: (title ~~ '%Buddhists, Shamans, and Soviets: Rituals of History in Post-Soviet Buryatia%'::text)
                     ->  Index Scan using works_unique on works referenced_work  (cost=0.70..8.67 rows=1 width=156)
                           Index Cond: (id = rw.referenced_work_id)
               ->  Index Scan using works_open_access_unique on works_open_access oa  (cost=0.70..8.67 rows=1 width=33)
                     Index Cond: (work_id = rw.referenced_work_id)
         ->  Index Scan using works_primary_locations_work_id_idx on works_primary_locations hv  (cost=0.57..8.54 rows=1 width=41)
               Index Cond: (work_id = rw.referenced_work_id)
 JIT:
   Functions: 24
   Options: Inlining true, Optimization true, Expressions true, Deforming true
(20 rows)
```

FIGURE 7.1: Query plan with wild card

In contrast, Query 2, which uses the = operator for an exact match, typically results in a simpler execution plan. With an indexed w.title column, the database optimizer can directly access the index to locate the rows with 'Buddhists, Shamans, and Soviets: Rituals of History in Post-Soviet Buryatia' as their w.title value, resulting in faster retrieval.

```
                                                                   QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=452.32..20598488.48 rows=1607 width=134)
   ->  Nested Loop  (cost=451.75..20584743.67 rows=1607 width=221)
         ->  Nested Loop  (cost=451.05..20570818.41 rows=1607 width=188)
               ->  Nested Loop  (cost=450.36..20556893.14 rows=1607 width=32)
                     ->  Index Scan using works_title on works w  (cost=0.82..677.22 rows=166 width=32)
                           Index Cond: (title = 'Buddhists, Shamans, and Soviets: Rituals of History in Post-Soviet Buryatia'::text)
                     ->  Bitmap Heap Scan on works_referenced_works rw  (cost=449.54..123516.44 rows=31619 width=64)
                           Recheck Cond: (work_id = w.id)
                           ->  Bitmap Index Scan on works_id_ref_idx  (cost=0.00..441.63 rows=31619 width=0)
                                 Index Cond: (work_id = w.id)
               ->  Index Scan using works_unique on works referenced_work  (cost=0.70..8.67 rows=1 width=156)
                     Index Cond: (id = rw.referenced_work_id)
         ->  Index Scan using works_open_access_unique on works_open_access oa  (cost=0.70..8.67 rows=1 width=33)
               Index Cond: (work_id = rw.referenced_work_id)
   ->  Index Scan using works_primary_locations_work_id_idx on works_primary_locations hv  (cost=0.57..8.54 rows=1 width=41)
         Index Cond: (work_id = rw.referenced_work_id)
 JIT:
   Functions: 18
   Options: Inlining true, Optimization true, Expressions true, Deforming true
(19 rows)
```

FIGURE 7.2: Query plan with exact match

In practice, when searching for exact matches in an indexed column, utilizing the = operator is recommended for optimal performance. However, if partial or pattern-based matching is required, it is essential to weigh the performance trade-offs and consider alternative indexing or search optimization techniques to mitigate potential latency associated with LIKE queries.

In conclusion, the choice between using LIKE with wildcards and the = operator depends on the specific use case and performance requirements. While LIKE offers flexibility for pattern-based searches, it may come at the cost of increased access times, especially when dealing with large datasets. Careful consideration of query

design and indexing strategies is essential to achieve the desired balance between flexibility and performance.