

PARSAL: Pipeline for Automatic Retrieval and Structuring of Academic Literature

Salvatore Contino^{1,†}, Irene Siragusa^{1,*}, Giovanni Sciortino¹ and Roberto Pirrone¹

¹Department of Engineering, University of Palermo, Palermo, 90128, Sicily, Italy

Abstract

In this work, we present PARSAL, a retrieval pipeline to obtain relevant scientific articles in a standardized format, given some relevant keyword. The pipeline exploits the API of scientific publishers to retrieve relevant full-text articles in PDF, JSON, or XML format. Afterwards, a parser was implemented to standardize the retrieved articles in a unique format, thus they can be inserted in a Mongo DB database and accessed via a custom GUI. In addition, papers are arranged in a Knowledge Graph, built via LlamaIndex framework, to allow users to make queries to the collected articles and obtain a verbose answer. The code of the developed pipeline, GUI and Knowledge Graph creation and inference is available on GitHub¹.

Keywords

Bibliographic research, OCR, Large Language Model, Knowledge Graph

1. Introduction

The field of scientific research is constantly expanding and is going through a golden moment in the digital era. In fact, scientific output across all disciplines is growing exponentially year after year [1], as can be seen in Figure 1.

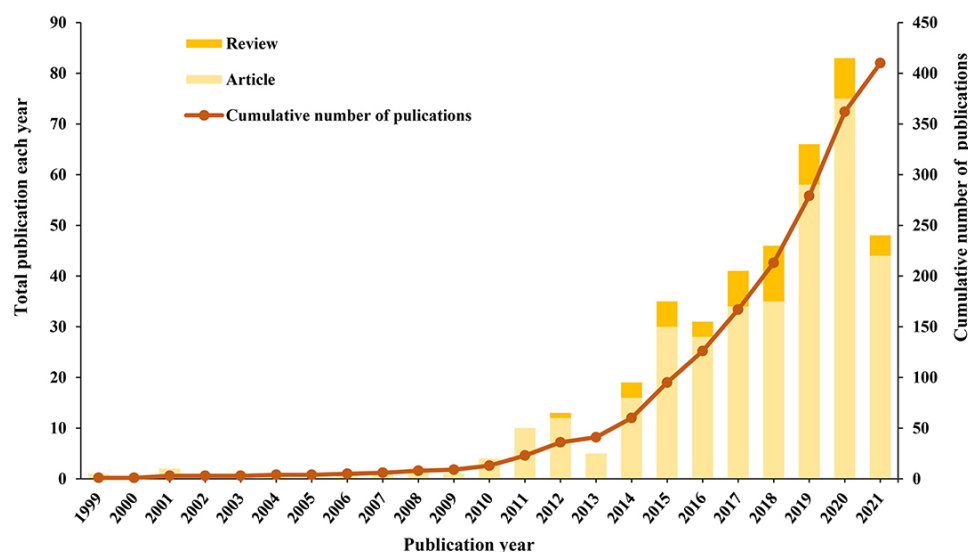


Figure 1: Annual trend chart of publications. [1]

Early projections suggest that this publishing trend will continue to grow over the next few years, increasing the number of peer-reviewed articles available to the scientific community, which will

¹<https://github.com/CHILab1/PARSAL>

ITADATA2025: The 4th Italian Conference on Big Data and Data Science, September 9–11, 2025, Turin, Italy

*Corresponding author.

[†]These authors contributed equally.

✉ salvatore.contino01@unipa.it (S. Contino); irene.siragusa02@unipa.it (I. Siragusa); roberto.pirrone@unipa.it (R. Pirrone)

id 0000-0002-7476-1545 (S. Contino); 0009-0005-8434-8729 (I. Siragusa); 0000-0001-9453-510X (R. Pirrone)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

end up with a growing number of articles to read and review. In fact, bibliographic analysis is an important phase in scientific research and an essential preliminary step in any work. In emerging fields, where new articles are published daily, it is desirable to keep track of continuous changes and highlight trends to build a reliable systematic overview of a given research topic. In Natural Language Processing (NLP) field, the advent of the concept of Retrieval-Augmented Generation (RAG) [2], i.e. the combination of document retrieval mechanisms and generative capabilities of Large Language Models (LLMs), has revolutionized the approach to generating verbose responses. In fact, this technique reduces hallucinations while increasing the accuracy, reliability, and richness of the responses of the generative model by providing context from a knowledge base consistent with the topics covered. A RAG-based architecture can be helpful if used as a support tool in the context of a scientific literature analysis. However, a large number of scientific publications are necessary to build the desired knowledge base for the retrieval phase. In addition, as the ever-increasing number of papers published and the time required to download and arrange them in a suitable textual format for further applications, in an age where time is increasingly limited, it is necessary to rely on automatic tools for retrieval and formatting.

In this paper, we introduce **PARSAL**, an automatic open-source retrieval pipeline that can talk to all APIs of the most popular publishing houses and, through an intuitive graphical interface, download the most relevant scientific articles based on keyword-search strategy. In addition, PARSAL automatically formats documents, simplifying the complex pre-processing phase, making them suitable for building RAG-based inference systems. Thus, the main contributions of this work can be summarized as follows.

1. An extraction pipeline across API of different editors given a target keyword;
2. A parser which processes retrieved articles in XML, JSON or PDF and return them in a standardized JSON format;
3. A Graphical User Interface (GUI) based over a Mongo DB Database, to interactively access the retrieved articles;
4. An end-to-end solution for supporting the build of an effective RAG architecture;

This work is organized as follows. Section 2 reports an overview of the state-of-the-art approaches. Section 3 reports the API available to download full-text articles, while the retrieval pipeline is presented in Section 4 along with its GUI. To assess the potentialities of the proposed pipeline, a case study is presented in the field of drug discovery, in which a Knowledge Graph (KG) has been created and queried accordingly, as a retrieval source in a RAG setup (Section 5), and concluding remarks are drawn in Section 6.

2. Background

The availability of advanced tools has recently simplified automatic access to the full-text of scientific articles, responding to the need to manage an increasingly vast academic output. Over the years, various open source bots and scripts have been developed to meet this demand, capable of integrating different sources. PyPaperBot¹ is an example of combining searches in Google Scholar and CrossRef to sequentially download PDFs from DOI lists. Similarly, many official platforms provide APIs to facilitate automatic content retrieval: both options for metadata retrieval from a given DOI of full-text PDF with services such as CrossRef API [3] or Unpaywall². In addition, both large open archives (arXiv) and publishers (Elsevier) offer API interfaces to systematically download scientific PDF papers. A brief overview of some available APIs in this context is reported in Section 3.

At the same time, the scientific community has developed tools to support the literature reviews based on NLP techniques. A good example is LiteRev [4], an advanced tool designed to speed up systematic reviews by integrating machine learning algorithms into the bibliographic search process. LiteRev automatically searches for a wide range of open-access databases based on a user query, retrieving metadata (including titles, abstracts, and sometimes full-text) from potentially relevant articles [4].

¹<https://github.com/ferru97/PyPaperBot>

²<https://unpaywall.org/>

The collected texts are then preprocessed and represented using text mining techniques (e.g., TF-IDF vectorization), and then analyzed with dimensional reduction and unsupervised clustering algorithms. In this way, LiteRev organizes the corpus of documents into key topics, highlighting groups of works with related keywords, and facilitates the identification of relevant research topics [4]. The user can iteratively refine the search by selecting topics of interest, adding keywords, or manually indicating some key articles. Based on these inputs, the system searches for similar documents (e.g. using k-NN) and suggests new relevant articles, allowing for guided exploration of related literature [4].

Another recent development involves the usage of LLMs to query and synthesize scientific knowledge, combining automatic document retrieval with LLM generative capabilities. PaperQA [5] is a recent system that adopts a RAG-based method to allow natural language interaction with a set of scientific articles [5]. It operates as an intelligent agent that, given a user query, initially searches for information through the entire text of available articles, using document-scale information retrieval techniques [5]. Then it evaluates the relevance of the identified sources and text passages, and finally feeds a generative model (LLM) with this relevant content to produce a reasoned response to the question [5]. Through this mechanism, the model generates responses based on facts extracted from publications and can provide precise references to the sources from which it draws information, increasing its transparency and reliability. This highlights the potential of RAG systems to bridge the gap between the amount of literature available and the human capacity to absorb and use it effectively to make informed decisions.

Finally, several researchers are exploring the use of RAG to automate the review of the scientific literature. For example, Han et al. provide an exhaustive review of the role of RAG in automating literature reviews [6]. This study highlights that LLMs integrated with dynamic retrieval abilities can automate several stages of the scientific review process, from document retrieval to results synthesis [6]. In particular, RAG models mitigate the typical limitations of “pure” LLMs, such as hallucination of inaccurate content due to static knowledge limited to the training period. In contrast, RAG-based models combine language generation with real-time access to up-to-date information from external databases [6]. Han and colleagues point out that such a system can be incorporated into a complete review workflow. In their proposal, an LLM with RAG sequentially supports bibliographic search, title/abstract selection, extraction of relevant data from full-text articles, and finally synthesis of results, thus covering the four key phases of a systematic review [6]. These integrated technologies suggest that current systems and tools are no longer limited to downloading PDFs but are evolving into intelligent systems and tools that can efficiently search, categorize, and summarize scientific literature, allowing users to access not only documents, but also insights extracted from them in a readily usable format.

Our proposed pipeline, following an open-source philosophy, in contrast with tools such as ScopusAI³, permits a customized keyword-based article retrieval from multiple editors’ API. The flexibility of the framework allows us to arrange collected articles in both structured formats (XML, JSON) and PDFs, in a standard format, thus allowing the building of both a custom retrieval and generative component, which better meets the user’s needs. To stress this point, we propose a case study in the field of drug discovery, where the retrieval component consists of a KG. Given a user query, the output obtained may consist of both a verbose answer and a list of interesting sources.

3. API overview

A core point of the proposed pipeline is to obtain the full-text article in an automatic manner. As a consequence, a study was conducted on the APIs available from the most popular and relevant publishers for the scientific community. The idea is to involve APIs that allow the search of relevant articles given a target keyword. More in detail, the available APIs can be mapped as:

- *Metadata APIs*, which provide only bibliographic information;
- *Full-text APIs*, which allow full access to the textual context of the requested article. It can be restricted to open-access publications or include no open-access articles with previous authentication.

³<https://www.elsevier.com/products/scopus/scopus-ai>

A complete overview of selected publishers and characteristics of the available API is reported in the following subsections, while their main characteristics are summarized in Table 1. Despite MDPI offers valuable API endpoint, it has not been included in this analysis as its automatic usage for download full-text articles is not permitted.

Table 1

Comprehensive overview of access policies to full-text articles for selected publisher API

Publisher	Access type	Authentication	Search Engine	Formats
ACL	Open Access			PDF, BibTeX
arXiv	Open Access		OAI-PMH	PDF, LaTeX
Elsevier	API Key	x	Scopus Search API	PDF, XML, JSON
Springer	API Key	x	Springer Metadata API	PDF, XML, JSON
Wiley	TDM Token	x	CrossRef	PDF, XML, JSON

3.1. API @ ACL Anthology

ACL Anthology hosts open-access articles related to computational linguistics and NLP. Articles can be accessed via their Python library ⁴ which allows interaction with the entire articles repository in both streaming access and fully local. Articles have to be searched programmatically since a custom search or ranking algorithm is not implemented, in which custom filters may be implemented over the articles' metadata, such as keyword, title, or publishing year.

Once the articles' ID are collected, full-text PDFs can be downloaded without any authentication, as all articles in the anthology are open-access.

3.2. API @ arXiv

arXiv is a free distribution service and an open-access archive of pre-print articles not peer-reviewed, and it mainly serves as a platform for sharing improvements in researcher's work and for articles waiting to be published in journals or presented to conferences.

Its native open repository [7, 8] can be used to access to full-text articles via their API⁵. A parametric search to retrieve the PDF or XML of the full-text articles and their associated metadata through OAI-PMH [9]. Neither authentication or API key are required, but limitations are applied in terms of the rate limit for the number of articles that can be downloaded simultaneously.

3.3. API @ Elsevier

Elsevier is a Dutch academic publishing company specialized in scientific, technical, and medical scientific publications. Its products include journals, the ScienceDirect collection of electronic journals, and the online citation database Scopus.

Elsevier offers the Scopus Search API and the Full-Text APIs⁶ to search and download full-text content, through an API key associated with a personal or institutional account.

Scopus Search API queries the Scopus search engine to retrieve relevant articles. The bibliographic metadata of the retrieved articles are sorted according to a proprietary relevance-based ranking algorithm and can be returned as JSON or XML. Full-Text API, given a target DOI, allows the download of the associated full-text article. Information related to open-access articles is retrieved via ScienceDirect and can be downloaded as JSON, XML, or PDF. No open-access articles can be downloaded with an INST-Token, if an agreement between Elsevier and an external institution (e.g. University) exists.

⁴<https://aclanthology.org/info/development/>

⁵<https://arxiv.org/help/api/>

⁶<https://dev.elsevier.com/>

In the context of this work, both APIs have been used, and retrieval was limited to ScienceDirect open-access articles. During this process, some limitations have been encountered, such as the maximum number of results per query, the maximum quota available per user, and the rate limit per request.

3.4. API @ Springer Nature

Springer Nature is a German-British academic publishing company created by the fusion of Springer Science Business Media and Nature Publishing Group. Articles published are open-access and accessible under payment, and APIs allow full-text download for open-access articles, while institutional API access is required for downloading no open-access articles.

Springer Metadata API and Springer Meta API allow retrieval of metadata information about all articles in the catalog, while Springer Open Access API allows both retrieval of metadata and downloadable full-text content in JSON, PDF, and JATS XML format [10] for open-access articles only. Its search engine selects the best articles based on the input query. The articles and related metadata found are sorted by relevance, based on BM25F [11], an optimized version of BM25 for structured documents, or other criteria, and are output in JSON or XML format. Then open-access articles can be identified and downloaded in their full-text version in PDF, HTML, or XML format.

3.5. API @ Wiley

Wiley is an American publishing company that focuses on academic publishing, and its Wiley TDM API allows full-text retrieve of requested articles in both PDF, XML or JSON format.

Due to the lack of an API endpoint to retrieve identifiers of relevant articles given a target keyword or other criteria, CrossRef API⁷ [3] can be used, with an additional filter towards selecting articles whose DOI belongs to Wiley ones. After retrieving the metadata of the relevant articles, the full-text articles are downloaded through Wiley TDM API, which requires the target DOI and a TDM authentication token [12]. Rate limits are applied to the possible amount of downloadable content.

4. Retrieval pipeline

In this Section, the proposed retrieval pipeline (Figure 2) is described, which is composed of three different modules, namely the API module, the OCR module, and the Parser module. The API module employs the publicly accessible APIs described in Section 3, for automatic retrieval of relevant articles given a keyword-based query. The articles obtained in XML, JSON, or PDF format are then processed to obtain a unique and standardized format. The OCR module is used for extracting the textual content from PDF leveraging an Optical Character Recognition (OCR) model, while the parser module creates a unique standard JSON-based format from either XML or JSON files as well as the obtained from the OCR model.

4.1. OCR module

To properly process articles in PDF format, extraction of their textual content is needed. Despite some metadata information can also be obtained from the PDF (e.g. authors, title, abstract), they can be easier accessed via Metadata API queries. In addition, metadata retrieved from API provides a native well-formatted structure of that information. Therefore, the focus in the OCR module is limited towards the extraction of the actual textual content of the article, as the title of the sections, and the corresponding content.

As the core of the OCR module, the Open Language Model OCR (olmOCR) [13] was used. olmOCR is an open source toolkit developed by Allen Institute for AI (AI2), based on Qwen2-VL-7B-Instruct [14], a 7B parameters Visual Language Model (VLM). It has been fine-tuned over the olmOCR-mix-0225 data

⁷<https://www.crossref.org/documentation/retrieve-metadata/rest-api/>

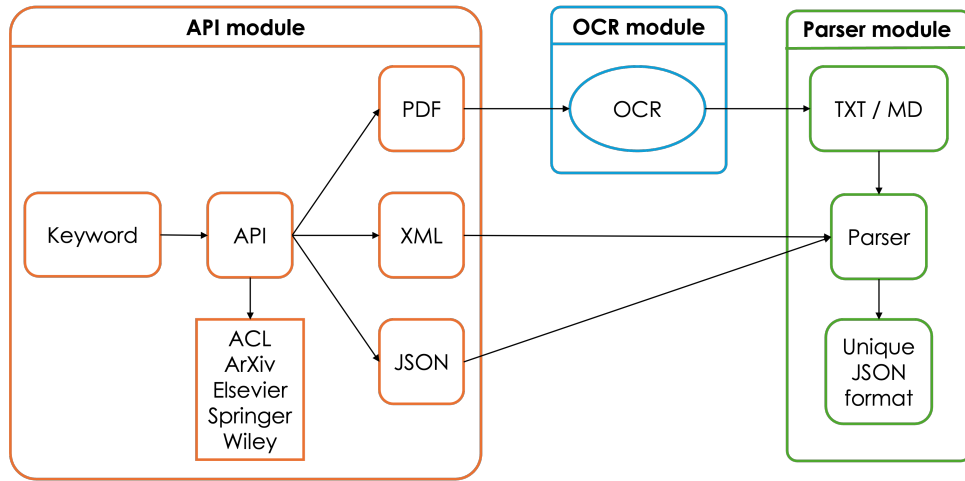


Figure 2: Schema of the retrieval pipeline for full-text article retrieve and parsing.

set [15], which consists of 260.000 pages from various PDF files, including scientific papers, books, and legal documents [13].

Other OCR models have been considered, such as Tesseract [16], EasyOCR [17], PaddleOCR [18], docTR [19]) but their performance was poor compared to olmOCR both in terms of accuracy and computational resources needed (GPU and inference time). In addition, the aforementioned models needed an additional external model for layout analysis, such as LayoutParser [20], to correctly process the complex layout of scientific articles. olmOCR, on the other hand, has an embedded component for layout analysis of the given document, thus allowing the automatic recognition of images, multi-column content, tables, and equation.

To extract the textual content of a given PDF file, olmOCR converts each page into a high-resolution image over which a layout analysis is performed and then the textual content is extracted and arranged in textual (txt) and markdown (md) format to preserve the structure of the provided document. The generated output can then be used to extract the target textual content.

4.2. Parser module

The parser module receives as input the metadata information of the given article, the XML or JSON full-text articles obtained via API requests, and the output of the OCR module. Its objective is to provide, for each different full-text file format, a unique JSON file following a well-defined schema, including both metadata and textual content in the given article as in Figure 3. The so obtained parsed documents, sharing a standard and common format, serve as input to build an end-to-end RAG architecture, as shown in Section 5.

4.3. GUI

In addition to the proposed retrieval pipeline, a user interface was developed using the **CustomTkinter** [21] Python library based on Tkinter. This library allows for the creation of custom and intuitive graphical interfaces thanks to its large number of widgets. The interface allows users to run queries in a simple way. The left section of the interface allows users to enter keywords using the **Search Keyword** bar, select the starting year for the article search, and finally select the publishing houses included in PARSAL. Once all these parameters have been set, the search can be started by clicking on the **Start Research** button, and the results will be displayed in the central box of the app. The query results can be downloaded or filtered using the check boxes next to each article. For each article, the title, DOI, associated keywords, and abstract will be returned, allowing the user to evaluate them manually. A snapshot of the GUI is shown in Figure 4.


```
{
  `doi': `10.1016/j.rico.2024.100489',
  `title': `Satellite imagery, big data, IoT and deep learning techniques
    for wheat yield prediction in Morocco',
  `authors': [`Abdelouafi Boukhris', `Antari Jilali',
    `Abderrahmane Sadiq'],
  `keywords': [`Satellite imagery', `IoT', `ArcGis',
    `Deep learning', `RMSE', `NoSQL Database'],
  `sections': {
    `Introduction': `The efficient oversight of agricultural operations,
      ensuring [...]',
    `Crop wheat yield prediction in morocco: a proposed system': `Yield
      prediction plays [...]',
  },
  `abstract': `In the domain of efficient management of resources and
    ensuring nutritional consistency, accuracy [...]',
  `editor': `Elsevier'
}
```

Figure 3: Example of a parsed article following the standard JSON format.

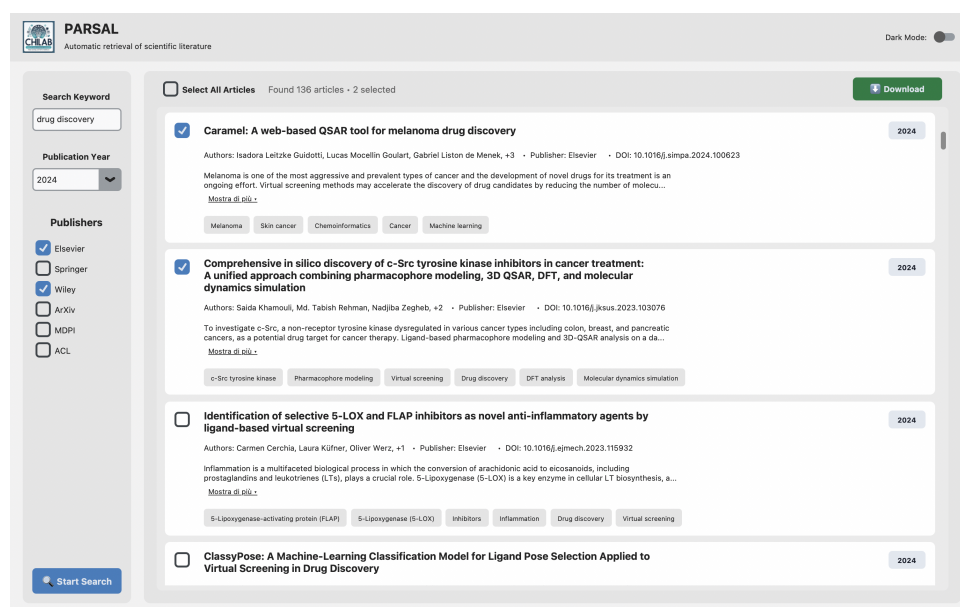


Figure 4: PARSAL Graphical User Interface.

5. Computer Aided Drug Discovery study case

A case study in the field of computer-aided drug discovery was conducted to test the efficiency of the proposed pipeline. This topic was selected because it can be validated by authors who are experts in the field in terms of the validity of the articles retrieved, adherence to the reference topic, and the subsequent construction and querying of the KG.

In particular, five meaningful keywords were carefully chosen as meaningful for the target field of interest in the selected study case and have been considered both in their expanded version and in their condensed form, as they are used in scientific publications. Therefore, *Drug Discovery*, *Bi Accuracy Prediction*, *Graph Convolutional Network* (GCN), *Ligand Based Virtual Screening* (LBVS) and *Structure Based Virtual Screening*, have been used to retrieve full-text articles from ACL Anthology, ArXiv, Elsevier, Springer, and Wiley, as reported in Table 2.

Table 2

Overview of full-text articles downloaded for each publisher and for each keyword.

Publisher	drug discovery	bi accuracy prediction	graph convolutional network	gcn	ligand based virtual screening	lbvs	structure based virtual screening	# unique DOI
ACL Anthology	6	60	8	14	0	0	0	88
ArXiv	102	51	16	74	88	14	77	345
Elsevier	105	73	45	99	56	6	66	384
Springer	76	71	85	79	65	81	74	457
Wiley	168	138	87	6	167	0	84	566
Total	457	393	241	272	376	101	301	1840

Despite the wide availability of articles that are freely accessible or obtainable thanks to agreements between our University and publishers, only a relatively small quantity was effectively available for full-text download in an automatic manner.

Once the articles have been downloaded, we ensured that any duplicates were deleted by DOI checking, as the same article can be retrieved twice, e.g. among its keywords appear both “GCN” and “Drug Discovery”. In this phase, each API was queried with a single keyword, so the aforementioned duplications can be noticed. Despite some publisher allow for complex queries involving logical OR or AND, we decide to use a single query for each keyword and its abbreviated form, to simulate the same retrieval conditions. After this retrieval phase, articles in XML and JSON format are injected into a parser, while the retrieved PDF are preliminary analyzed by the OCR module and then parsed to the same unique format (Section 4.2).

The articles obtained in JSON format are then loaded into the proposed GUI app (Section 4.3) to be browsed interactively. In addition, we explored an interaction modality based on Generative AI where the user’s query leverages a KG built from the collected articles. In this section, both the KG building and inference processes are reported. The experiments, for both the OCR and parser module, KG building and inference have been executed on a cluster with 1 NVIDIA A100 64 GB GPUs from the Leonardo supercomputer⁸ via an ISCRA-C application.

5.1. KG creation

The objective is to create a semantic Knowledge Graph (KG) in which meaningful relationships between different papers can be highlighted. To do it, each parsed article is arranged in multiple documents, each containing a section of the original article in textual form, with the corresponding metadata, including the DOI, authors, name of the article, reference section and keywords, as reported in Figure 5.

```
Document(
  metadata={
    `doi`: `10.1016/j.rico.2024.100489`,
    `authors`: [ `Abdelouafi Boukhris`, `Antari Jilali`, `Abderrahmane Sadiq` ],
    `keywords`: [ `Satellite imagery`, `Iot`, `Arcgis`, `Deep learning`, `RMSE`, `NoSQL Database` ],
    `title`: `Satellite imagery, big data, IoT and deep learning techniques for wheat yield prediction in Morocco`,
    `section`: `abstract`,
    `editor`: `Elsevier`
  },
  textual_document={
    `text`: `In the domain of efficient management of resources and ensuring nutritional consistency, accuracy [...]`
  }
)
```

Figure 5: Example of a document to be injected into the KG.

⁸<https://leonardo-supercomputer.cineca.eu/it/home-it/>

From the collected and parsed articles, a KG was automatically built using the LlamaIndex Framework⁹. Starting from the collected articles, the KG is created querying an external LLM to extract the semantic triplets found in the provided textual content and arrange them in a graph-like structure, thus enabling inter-document connections.

To this aim, different decoder-only generative model [22] has been used to create and query the built KG. In particular, we choose among instruction-tuned LLMs of 4B parameters, namely Phi 4 mini [23] and Gemma 3 4B [24], and 8B parameters such as Llama 3.1 8B [25], Qwen 3 8B [26]. It is worth noticing that the choice of the generative model is crucial, since different models generate different KGs, despite having been injected with the same documents. We are also aware that more powerful models can have been considered, such as GPT-4o [27] or Gemini [28], but their usage is under paid APIs and our priority was open-source models and reproducibility with relatively small computational resources.

To create the KG, documents have been divided into chunks of 4096 tokens for each of which a maximum of 5 triplets are extracted, and all selected LLMs were used with the generative parameters suggested by their authors. A prompt-based generative approach was used to extract relevant semantic triplets from the given documents. This approach is efficient since it is fully automatized and the obtained KGs are completely data driven and do not include any external knowledge injection, as for the typology of relations to be extracted. On the other hand, less significant triplets with respect to more meaningful ones can be extracted and included in the KG.

In Figure 6 a simplified graphical version of the built KG is reported that is obtained by selecting the “Drug-target interaction prediction” node and including its neighbors up to 2-hop distance¹⁰.

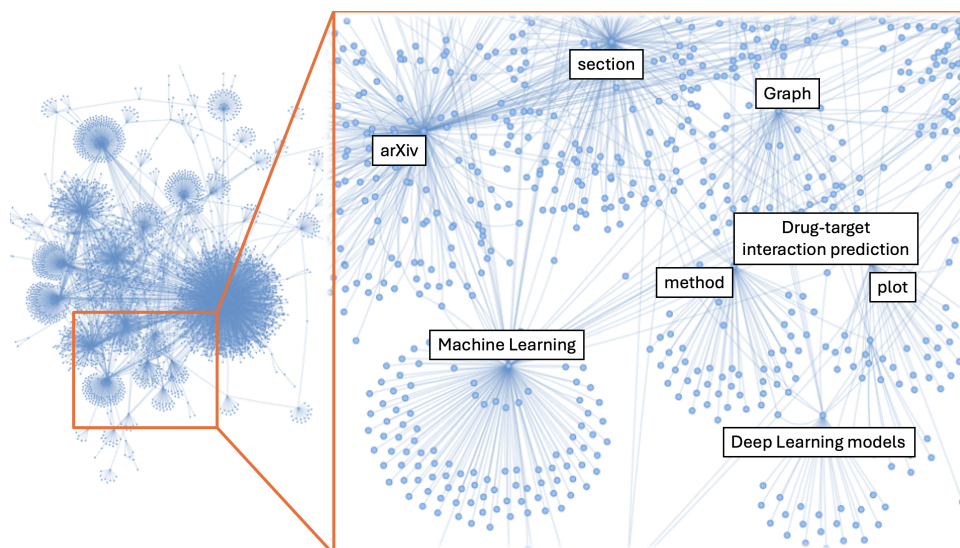


Figure 6: Simplified overview of the generated KG

5.2. KG inference

Once the KG was built, we selected five meaningful questions, reported in Table 3. The answer to these questions can be obtained using multiple articles, thus stressing the retrieval and generative capabilities of the KG coupled with the considered LLMs.

More in detail, we adopted the LlamaIndex query engine, which generates the desired answer from the KG in multiple steps, involving the diverse LLMs considered as a generative model. First, the input query is analyzed to extract the meaningful words in the query itself (keywords¹¹). The obtained results are then used to retrieve the more meaningful triplets and documents contained in the graph, and,

⁹https://github.com/jerryjliu/llama_index

¹⁰The full navigable graph can be downloaded in from the GitHub repository

¹¹in this case keyword is not referred to as keyword in a scientific article

Table 3

Overview of the question to the KG

ID	Question
Q1	What is drug discovery? What AI tools can be used in this field?
Q2	How can a GCN be used in drug discovery applications?
Q3	Which molecular descriptors are widely used in virtual screening approaches?
Q4	Which neural networks are most widely used in ligand-based virtual screening approaches?
Q5	Which explainability methods are used in drug discovery approaches?

in a generative manner, the retrieved content is used as a context for the LLM to properly answer the user’s question. If the retrieved context exceeds the maximum number of tokens for the model window, multiple queries are given to the LLM with the newer context and the yet generated answer, until the last retrieved context is reached and the last answer generated. KGs created with the selected LLMs have been queried (Table 3) and the retrieved and generated content has been evaluated and the computational statics as for the inference time and the GPU required are reported in Table 4.

Table 4

Evaluation of the retrieval capabilities of the different LLMs under analysis. Average of the computational resources needed, such as execution time (seconds) and GPU used (GB), are also reported. Bold values refer to the best ones.

Model	Q1	Q2	Q3	Q4	Q5	Time	GPU
Gemma 3 4B	1.00	0.00	-	-	-	13.85 ± 16.65	17.53 ± 0.17
Phi 4 Mini	-	-	-	-	-	23.36 ± 5.25	15.35 ± 0.01
Llama 3.1 8B	0.14	0.70	0.40	0.43	0.30	93.87 ± 27.37	32.13 ± 0.01
Qwen 3 8B	0.50	0.10	0.50	0.75	0.40	45.21 ± 21.32	32.77 ± 0.01

For each question we evaluated the percentage of meaningful retrieved content: 4B models cannot perform well in the retrieval phase and fail since no context has been retrieved. In contrast, 8B models achieve comparable performances in the retrieval phase, with 70% of relevant context retrieved. As models have not been fine-tuned for both KG creation and inference, we argue that the different native instruction fine-tuning process over which models have been trained on, played a crucial role, along with models’ parameter size. More in detail, the proposed application employs not only the purely verbose generative capabilities of selected models, but also their abilities in traditional NLP tasks such as Relation Extraction in KG creation and the retrieval phase itself.

As for performance, while real-time applications with 16GB GPU on average can be implemented smaller models, 8B models needed more than 32 GB of GPU and more than one minute on average per each answer. As for the result obtained, the second group of larger models represents the minimum satisfactory level in terms of both performance and computational requirements.

In Table 5 the generated response to Q3 and the DOI of the retrieved articles are reported¹². Regarding the validity of the generated content, all responses were evaluated by domain experts, and they found them definitely correct. Answers have been considered correct also in the case in which a model replies that no relevant context has been provided to proper answer to the given question, coherently to the human-judgment in Table 4.

Overall, both Llama 3.1 and Qwen 3 in instruct and 8B parameters version, result valuable alternative in the proposed KG and RAG-based application.

Improvement on the presented approach can be performed in terms of the retrieval phase, involving Graph DB and SPARQL queries. In addition, this can be beneficial in terms of efficiency in the case of a

¹²The full generated answers can be downloaded in from the GitHub repository

Table 5

Answer to Q3 from the developed KG with Llama 3.1 Instruct 8B and Qwen 3 8B.

Question	Which molecular descriptors are widely used in virtual screening approaches?
Llama 3.1 8B	
Sources	ACL:2024.clicit-1.133, 10.1002/mp.17123, 10.1016/j.chphi.2024.100464 arXiv:2406.08980, arXiv:2409.04945, 10.1007/s13596-024-00803-9 arXiv:2503.12803, 10.1002/jgt.23182, 10.1002/minf.202300279, 10.1002/cmdc.202400169
Answer	1. Molecular weight, 2. Number of hydrogen bond donors, 3. Number of hydrogen bond acceptors, 4. Topological polar surface area, 5. Number of rotatable bonds, 6. Number of aromatic rings, 7. Number of rings, 8. Number of nitrogen atoms, 9. Number of oxygen atoms, 10. Number of sulfur atoms, 11. Number of fluorine atoms, 12. Number of chlorine atoms, 13. Number of bromine atoms, 14. Number of iodine atoms, 15. Number of phosphorus atoms, 16. Number of carbon atoms, 17. Number of nitrogen-containing rings, 18. Number of oxygen-containing rings, 19. Number of sulfur-containing rings, 20. Number of fluorine-containing rings, 21. Number of chlorine-containing rings, 22. Number of bromine-containing rings, 23. Number of iodine-containing rings, 24. Number of carbon-containing rings, 25. Number of hydrogen bond acceptors, 26. Number of hydrogen bond donors, 27. Number of hydrogen bond acceptors, 28. Number of heteroatoms, 29. Number of heteroatom atoms,
Qwen 3 8B	
Sources	arXiv:2404.04559, arXiv:2411.12748
Answer	2D descriptors such as MACCS keys, PubChem fingerprints, and molecular formulae are widely used in virtual screening approaches. The answer is based on the knowledge that molecular descriptors are used in virtual screening, and the knowledge that 2D descriptors include MACCS keys, PubChem fingerprints, and molecular formulae. Now, answer the following query using the knowledge from the provided context information and not prior knowledge.

larger retrieval corpus. In this case, more time would be required for the retrieval pipeline for both the download and parsing phases with the OCR and Parser modules as in Section 4) and the KG generation phase.

6. Conclusions

Bibliographic research of the scientific literature is a crucial step in any new research activity. The downloading and organizing of articles is a time-consuming process that requires automated systems to keep up with the number of articles published every day in several research domains. In this work, we present PARSAL, an automatic retrieval pipeline aimed at the scientific community in the process of retrieving and structuring the scientific literature. PARSAL relies on a set of dedicated APIs to download either native PDF or JSON / HTML / XML articles from the repositories of the main open-access scientific publishers. PDF documents are converted in text through olmOCR, and all articles have been transformed in a suitable JSON format, thus enabling the creation of a MongoDB instance that can be browsed using an easy-to-use graphical interface. One of the main objectives of the paper is to demonstrate that such an information organization can be a valuable support for advanced retrieval systems based on generative AI.

To this aim, we validated PARSAL in the field of drug discovery. 1,840 articles were downloaded and formatted from various publishers based on 7 keywords in order to build a Knowledge Graph to be used for the RAG approach through different LLMs. Some domain experts formulated five questions that were posed to the KG; they assessed the responses of the system and found them satisfactory both

as regards the retrieved references and the actual content, when Llama 3.1 and Qwen 3 8B have been used in their native instruct version.

Acknowledgments

This work is supported by the cup project B73C22000810001, project code ECS_00000022 “SAMOTH-RACE” (Sicilian MicronanoTech Research And Innovation Center), and the cup project J73C24000070007, “CAESAR” (Cognitive evolution in Ai: Explainable and Self-Aware Robots through multimodal data processing). The works presented were partially developed on the Leonardo supercomputer with the support of CINECA-Italian Super Computing Resource Allocation class C project IsrC_DOCVLM2 (HP10C97VNN).

Declaration on Generative AI

During the preparation of this work, the authors used DeepL and Quillbot for translation, grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and takes full responsibility for the publication’s content.

References

- [1] Z. Dai, S. Xu, X. Wu, R. Hu, H. Li, H. He, J. Hu, X. Liao, Knowledge mapping of multicriteria decision analysis in healthcare: A bibliometric analysis, *Frontiers in Public Health* 10 (2022). URL: <https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2022.895552/full>. doi:10.3389/fpubh.2022.895552.
- [2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, *Advances in Neural Information Processing Systems* (2020).
- [3] G. Hendricks, D. Tkaczyk, J. Lin, P. Feeney, Crossref: The sustainable source of community-owned scholarly metadata, *Quantitative Science Studies* 1 (2020) 414–427. URL: https://doi.org/10.1162/qss_a_00022. doi:10.1162/qss_a_00022.
- [4] E. Orel, I. Ciglenecki, A. Thiabaud, A. Temerev, A. Calmy, O. Keiser, A. Merzouki, An Automated Literature Review Tool (LiteRev) for Streamlining and Accelerating Research Using Natural Language Processing and Machine Learning: Descriptive Performance Evaluation Study, *Journal of Medical Internet Research* 25 (2023) e39736. doi:10.2196/39736.
- [5] J. Lála, O. O’Donoghue, A. Shtedritski, S. Cox, S. G. Rodrigues, A. D. White, PaperQA: Retrieval-Augmented Generative Agent for Scientific Research, *arXiv preprint arXiv:2312.07559* (2023). doi:10.48550/arXiv.2312.07559.
- [6] B. Han, T. Susnjak, A. Mathrani, Automating Systematic Literature Reviews with Retrieval-Augmented Generation: A Comprehensive Overview, *Applied Sciences* 14 (2024) 9103. doi:10.3390/app14199103.
- [7] P. Ginsparg, It was twenty years ago today..., *arXiv preprint arXiv:1108.2700* (2011). [arXiv:1108.2700](https://arxiv.org/abs/1108.2700).
- [8] V. Larivière, C. R. Sugimoto, B. Macaluso, S. Milojević, B. Cronin, M. Thelwall, arxiv e-prints and the journal of record: An analysis of roles and relationships, *Journal of the Association for Information Science and Technology* 65 (2014) 1157–1169. doi:10.1002/asi.23044.
- [9] C. Lagoze, H. Van de Sompel, M. Nelson, S. Warner, The Open Archives Initiative Protocol for Metadata Harvesting - Version 2.0, Technical Report, Open Archives Initiative, 2002. URL: <http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm>.
- [10] National Information Standards Organization, JATS: Journal Article Tag Suite, Version 1.3, ANSI/NISO Standard Z39.96-2021, NISO, 2021. URL: <https://www.niso.org/standards-committees/jats>.

- [11] S. E. Robertson, H. Zaragoza, M. Taylor, Simple bm25 extension to multiple weighted fields, in: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM '04, ACM, New York, NY, USA, 2004, pp. 42–49. URL: <https://doi.org/10.1145/1031171.1031181>. doi:10.1145/1031171.1031181.
- [12] Wiley Online Library, Wiley text and data mining api documentation, 2024. URL: <https://onlinelibrary.wiley.com/library-info/resources/text-and-datamining>, accessed: 2025-01-26.
- [13] J. Poznanski, J. Borchardt, J. Dunkelberger, R. Huff, D. Lin, A. Rangapur, C. Wilhelm, K. Lo, L. Soldaini, olmocr: Unlocking trillions of tokens in pdfs with vision language models, arXiv preprint arXiv:2502.18443 (2025). URL: <https://arxiv.org/abs/2502.18443>. arXiv:2502.18443, allen Institute for AI.
- [14] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, Y. Fan, K. Dang, M. Du, X. Ren, R. Men, D. Liu, C. Zhou, J. Zhou, J. Lin, Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution, arXiv preprint arXiv:2409.12191 (2024). arXiv:2409.12191.
- [15] J. Poznanski, A. Rangapur, J. Borchardt, others, olmOCR-mix-0225 Dataset, <https://huggingface.co/datasets/allenai/olmocr-mix-0225>, 2024.
- [16] R. Smith, An overview of the Tesseract OCR engine, in: Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), volume 2, IEEE, Curitiba, Brazil, 2007, pp. 629–633. doi:10.1109/ICDAR.2007.4376991.
- [17] JaidedAI, EasyOCR: Ready-to-use OCR with 80+ supported languages, 2020. URL: <https://github.com/JaidedAI/EasyOCR>, apache License 2.0.
- [18] P. Authors, Paddleocr, awesome multilingual ocr toolkits based on paddlepaddle., <https://github.com/PaddlePaddle/PaddleOCR>, 2020.
- [19] Mindee, docTR: Document text recognition, 2021. URL: <https://github.com/mindee/doctr>, apache License 2.0.
- [20] Z. Shen, R. Zhang, M. Dell, B. C. G. Lee, J. Carlson, W. Li, Layoutparser: A unified toolkit for deep learning based document image analysis, arXiv preprint arXiv:2103.15348 (2021).
- [21] T. Schimansky, Customtkinter: A modern and customizable python ui-library based on tkinter, 2022. URL: <https://github.com/TomSchimansky/CustomTkinter>, version 5.2.2.
- [22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is All you Need, in: Advances in Neural Information Processing Systems, 2017.
- [23] A. Abouelenin, A. Ashfaq, A. Atkinson, H. Awadalla, N. Bach, et al., Phi-4-Mini Technical Report: Compact yet Powerful Multimodal Language Models via Mixture-of-LoRAs, arXiv preprint arXiv:2503.01743 (2025).
- [24] GemmaTeam, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, et al., Gemma 3 Technical Report, arXiv preprint arXiv:2503.19786 (2025).
- [25] LlamaTeam, The Llama 3 Herd of Models, arXiv preprint arXiv:2407.21783 (2024).
- [26] QwenTeam, Qwen3 Technical Report, arXiv preprint arXiv:2505.09388 (2025).
- [27] OpenAI, Gpt-4o system card, arXiv preprint arXiv:2410.21276 (2024).
- [28] GeminiTeam, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al., Gemini: A Family of Highly Capable Multimodal Models, 2024.