ECS736U/ECS736P - Information Retrieval

Designing a Specialised Search Engine for Machine Learning Academic Journals: An Information Retrieval Project

By

Stef Abolade

Introduction

In the rapidly advancing field of computer science, the surge in academic research, particularly within machine learning, highlights the need for specialised search engines. Unlike general search platforms that often fail to navigate the dense and complex landscape of academic publications effectively, specialised engines offer advanced search capabilities tailored to meet the unique demands of academic research.

The challenge of efficiently locating relevant machine-learning literature comes from the field's exponential growth, which leads to an increase in publications each year (Rajaraman & Ullman, 2011). This rise, indicative of the field's progress, poses a significant hurdle in identifying the most current and impactful studies. The volume of available research and the rapid pace of technological advancements highlight the importance of timely access to the latest findings for researchers and students (Manning et al., 2008).

This project proposes the development of a search engine tailored explicitly for retrieving academic journals in machine learning. We aim to overcome the barriers encountered in academic research by providing a focused, efficient, and user-friendly platform for accessing machine learning literature. By targeting machine learning, the search engine will use indexing, query processing, and relevance ranking algorithms to ensure precise and relevant search results, enhance its users' research process and productivity, and support ongoing growth and innovation within the field.

Embarking on this project, we aim to bridge a critical gap in academic research tools, contributing to efforts to simplify the discovery and use of scholarly information in the fast-paced, ever-evolving realm of machine learning.

Aim

The primary aim of our project is to develop a specialised search engine designed for the retrieval of academic literature in the field of machine learning. This addresses the challenges the machine learning community faces in identifying relevant publications in a fast-growing volume of research (Boeker et al., 2013). By integrating indexing, preprocessing, and the BM25 algorithm for query processing, our search engine aims to improve the efficiency, relevance, and accuracy of document retrieval. Unlike existing general-purpose search engines, which offer broad access and often lack the precision and specialised features for machine learning research, our project aims to bridge this gap in academic research tools (Gusenbauer & Haddaway, 2020). Offering a solution that is tuned to the needs of machine learning scholars. In addition to improving search functionality, an objective is to design a user-friendly interface that simplifies the research workflow. This aims to make the discovery of machine learning literature more accessible and tailored to the community's needs, improving the overall research experience and facilitating accessible access to essential resources. We will use a holistic evaluation approach to ensure the project is successful and meets the specific demands of machine learning academic research. Key performance metrics will be used to assess the search engine's effectiveness, including precision, recall, and user satisfaction. These measures will help ensure actionable insights, which will drive the continuous refinement and optimisation of the search engine.

Through this approach, our project aims to support ongoing research, contribute to academic progress, and stimulate innovation within the machine learning field.

Task

The following tasks outline our approach to developing a specialised search engine for machine-learning academic literature:

- ***Data Collection***: We will gather a dataset with academic journals and other relevant documents from the web, creating a comprehensive database for our search system.
- ***Document Cleaning and Preprocessing***: The collected documents will be cleaned and preprocessed using natural language processing techniques such as tokenisation, stemming and stop-word removal. This process aims to enhance data quality, ensuring the text is in a format suitable for analysis and query matching.
- ***Indexing***: We will select data structures and a retrieval model aligned with the search engine's requirement using indexing techniques. The preprocessed data will be systematically organised to prioritise fast and efficient retrieval, effectively linking user queries with relevant documents.

  Query Processing with BM25: Using the BM25 algorithm, we will process tokenised queries to generate a list of documents alongside their relevance and accuracy scores. Parameters will be adjusted to reflect the dataset's characteristics, ensuring search results are accurately ranked by relevance.

- ***User Interface Design***: A user-friendly interface will be developed to simplify the search experience. This interface will support easy query input and search functionalities and display the search outcomes to users.
- ***Evaluation and Optimisation***: Using manually labelled ground truth data, we will assess the search engine's performance through key metrics such as precision, recall and user satisfaction. The insights gained will guide adjustments and improvements to the search engine, improving query effectiveness and the relevancy of returned documents.
- ***Documentation and Reporting***: We will document the development process, the architecture of the search engine, and the results of our evaluation phase. This will provide insights and recommendations for future project enhancements.

Dataset

The dataset is central to developing our specialised search engine for machine learning academic literature. It supports the development and evaluation phases. It must reflect the diversity and depth of the machine learning field, enabling efficient and accurate document retrieval.

The dataset includes peer-reviewed academic journals, conference papers, and related academic work across various subfields of machine learning. It includes but is not limited to, supervised and unsupervised learning, reinforcement learning and deep learning. This will be sourced from reputable academic databases and digital libraries like JSTOR, Google Scholar, IEEE Xplore, ACM Digital Library, PubMed, Scopus and Web of Science, ensuring a diverse collection of machine learning literature.

For preprocessing, the dataset undergoes several steps:

Tokenisation: Breaking down text into individual words or terms

Stemming: Reducing words to their root form

Stop-word Removal: Removing common words that add little value to retrieval efforts.

These steps are vital in standardising the dataset and improving consistency, which is essential for effective indexing and retrieval.

To ensure the search engine's evaluation is comprehensive, we will identify representative queries that cover a wide spectrum of topics within machine learning. This includes broad queries like "machine learning fundamentals" to assess the engine's capability in providing foundational content, specific request queries such as "deep learning techniques for image classification" to test retrieval of specific documents and queries like "supervised vs. unsupervised algorithms" for evaluating its ability to source comparative studies. This approach enables us to simulate a range of searches. Ensuring that our search engine can match user queries across varying levels of specificity and that the evaluation reflects real-world search scenarios.

Our evaluation methodology combines binary relevance labelling, classifying documents as 'relevant' or 'irrelevant' based on their alignment with predetermined search queries. This simplifies the creation of a quantifiable ground truth. We will integrate performance metrics such as precision (the proportion of retrieved documents that are relevant), recall (the proportion of relevant documents that are retrieved) and the F1 score (a mean of precision and recall) to assess retrieval effectiveness quantitatively. Additionally, we will collect user satisfaction feedback from our team, representing the broader machine learning community and assessing the search engine's usability and overall user experience. Based on the outcomes of these evaluations, we will systematically make improvements to the search engine, including the BM25 algorithm, indexing strategies, preprocessing techniques and the user interface design. Each adjustment will be followed by a re-evaluation to measure the impact of modifications made, ensuring the search engine aligns with our retrieval efficiency, relevance and user satisfaction objectives. Documentation of each adjustment, its rationale, and its impact will provide a record of the search engine's development and alignment with our project goals.

A general architecture of the search engine

The proposed architecture illustrated below gives an overview of how the system will be split into three main components. Each component section will describe the details.

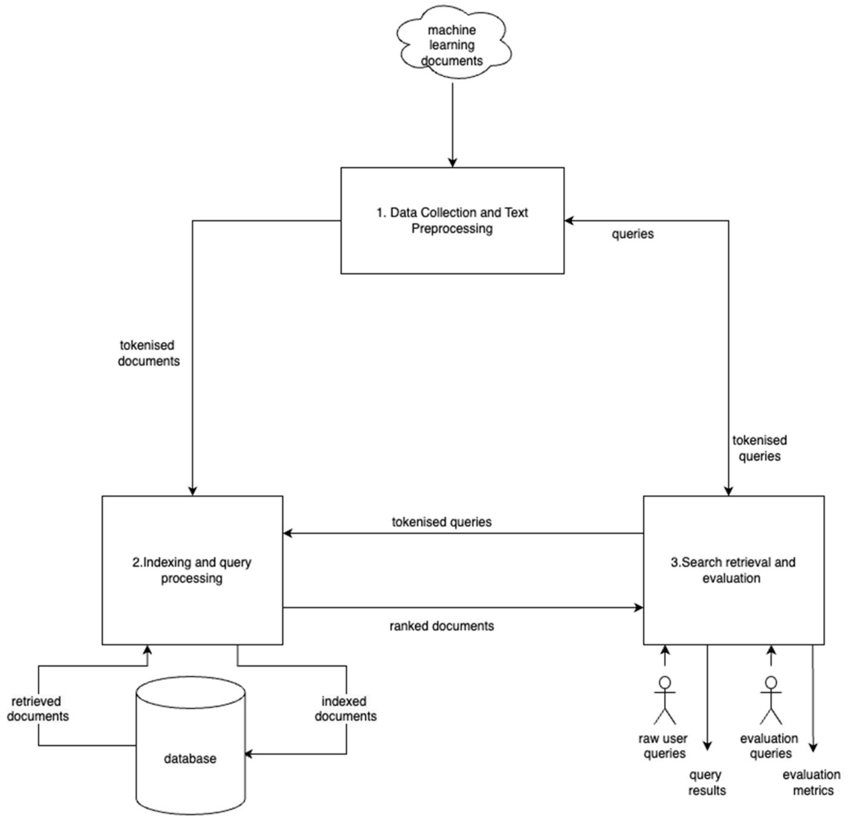


Figure 1: Architecture of the Machine Learning Literature Search Engine

Component 1 – Data Collection and Text Preprocessing

The first component is responsible for web crawling and retrieving a collection of suitable documents for our Machine learning academic journal search system. Additionally, it will be tasked with cleaning, refining, and preprocessing those documents using a selection of natural language processing techniques. The same methods will be applied to incoming queries to maintain a level of consistency of text processing across queries and documents.

Component 2 – Indexing and Query Processing

The second component is tasked with indexing the incoming tokenised documents according to the chosen retrieval model by providing an indexing function which will be called by component 1. It will then store these documents in a database or a folder for future retrieval. This will be enabled by a retrieval function which will return a list of documents and their retrieval model matching score upon receipt of a tokenised query.

Indexing Tokenised Documents:

- Tokenisation: Documents received from Component 1 are already preprocessed, including tokenisation, which breaks down text into smaller units like words or phrases. This step is crucial for indexing as it simplifies the document content into searchable elements.
- Indexing Function: This is the core function of Component 2. It takes the tokenised documents and organises them in a way that makes them easily searchable. The indexing function maps document identifiers (such as titles, authors, or unique IDs) to their corresponding tokens. This process involves creating an inverted index, a data structure that lists each token and which documents contain it. The goal is to facilitate a quick lookup of documents containing specific tokens.
- Storage: After indexing, the documents are stored in a database or a folder. This storage not only holds the original documents but also the indexes created, allowing for efficient retrieval based on the tokens.

Query Processing

- Receiving Tokenised Queries: Just like documents, queries undergo preprocessing and tokenisation before being submitted to Component 2. This ensures consistency in how documents and queries are treated, improving the relevancy of search results.
- Retrieval Function: Upon receiving a tokenised query, the retrieval function utilises the indexing structure to find documents that match the query tokens quickly. It then applies the chosen retrieval model to calculate a matching score for each document.
- Returning Results: The system sorts the documents based on their matching scores and returns a list of documents, often with a limit on the number of results or sorted by relevance. This list includes document identifiers, titles, authors, and abstracts.

Component 3 – Search Retrieval and Evaluation

The final component contains the search retrieval and evaluation systems. The former subsystem enables a user to input a query, which gets sent to component 1 for preprocessing before it gets passed onto component 2 via the above-mentioned retrieval function. Upon receipt of the documents deemed relevant, it will rank the items by their scores, if necessary, before returning the results to the user. The latter subsystem is tasked with subsequently evaluating the quality of the returned ranking according to the chosen evaluation metrics and summarising the statistics before returning this to the user upon an evaluation query request. This component is responsible for a smooth user experience by providing information and help flags [-I, -h] to help with user input as is standard with most Python command line user interfaces.

Design

This subcomponent will have three classes. Firstly, it will have a user interface class which provides a way to query the system. Secondly, it will have a ranking class which interacts with the indexer component to retrieve and rank documents. And finally, it will have an evaluation class which calculates the evaluation metrics.

Tasks

- Create a class for user input which takes in query prompts.
    - Enable and parse user queries.
    - Create a way to differentiate between search queries and evaluation prompts.
    - Provide helpful prompts upon erroneous user inputs.
    - Pass queries onto text preprocessing sub-component to tokenise queries.
- Create a class to interact with the indexer main component to retrieve documents.
    - Pass tokenised queries onto indexer components to receive scored documents.
    - Create a ranking function which ranks based on scores.
    - Create a function which outputs the ranked lists of documents to the user in a comprehendible manner.
- Create a class to evaluate the rankings based on an evaluation query provided by the user.
    - Upon receiving an evaluation prompt, call above class to receive ranking for prompt.
    - Call indexer component to provide ground truths from database for returned documents.
    - Based on ground truths and ranking, calculate chosen metrics
    - Create a function to output metrics in a comprehendible manner to be returned to user.

Retrieval model

The BM25 algorithm is a widely used ranking function in the field of Information Retrieval (IR), particularly for search engine document ranking. It belongs to the family of probabilistic information retrieval models and is an extension of the binary independence model. The BM25 algorithm calculates the relevance of documents to a given search query, facilitating the retrieval of the most pertinent documents from a large collection.

TF (Term Frequency) Component: BM25 incorporates the frequency of query terms in the document, acknowledging that documents containing a query term more frequently are likely more relevant. However, it introduces a saturation point to prevent excessively high term frequency from disproportionately increasing a document's relevance score.

IDF (Inverse Document Frequency) Component: The algorithm considers the rarity of terms across the document collection, giving higher weight to rarer terms. This is based on the principle that rare terms contribute more to the uniqueness and specificity of a document's content, making them more valuable for relevance scoring.

Length Normalisation: BM25 adjusts scores based on document length, mitigating the bias towards longer documents by normalising the impact of document length on relevance. This ensures that both short and long documents are fairly evaluated for their relevance to the query.

These features collectively enable BM25 to effectively rank documents by relevance in a manner that is responsive to both the frequency and distribution of query terms within and across documents, as well as to the inherent variability in document lengths within a collection. The flexibility offered by its tuning parameters further enhances its adaptability, making BM25 a versatile and powerful tool in the field of information retrieval.

The tool(s) you will be using to build the search engine

To build our search engine, we have selected tools that align with our project's technical requirements and allow for operational efficiency. These include the following:

- BeautifulSoup is a Python library that analyses HTML and XML documents. It extracts data from web documents and is vital for gathering content from diverse academic sources (Hajba, 2018).
- Whoosh was selected as a pure Python search engine library. It serves as our primary library and provides simple and direct support for the Okapi BM25 algorithm, which is essential for organising information within the search engine's dataset for our retrieval model (Chaput, 2012).
- FastAPI is applied to create the web interface of our search engine. It allows for high performance while being a simple tool for creating web applications. It also has the added benefits of automatic request validation and interactive API documentation (Tiangolo, 2018).
- Pandas and NumPy libraries are key for data handling and analysis. Pandas offer high-level data structures, while NumPy provides complex numerical calculations. Together, they are crucial for preprocessing and organising our dataset (Miller, 2018).
- Natural Language Toolkit (NLTK) is used for its libraries, which can process textual data and support the development of algorithms for search engines, which is key for preprocessing data (Hardeniya et al., 2016).

The project uses GitHub for collaborative coding, supporting team coordination, and managing the project (Loeliger & McCullough, 2012).

Teamwork and Responsibilities

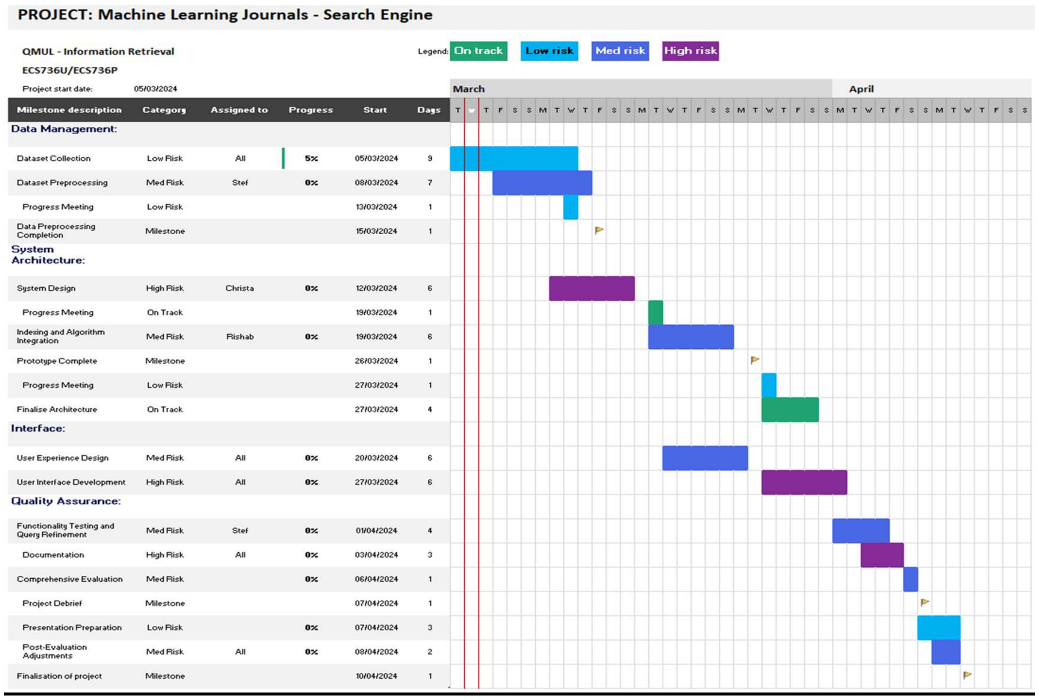
Our project uses a collaborative approach, with each team member having responsibilities aligned with their expertise. The tasks are designed to leverage individual strengths while encouraging accountability. Our data collection efforts are a joint responsibility to collect over 100 relevant academic journals. Stef will be responsible for leading the preprocessing phase, ensuring the integrity and uniformity of data for retrieval purposes. While all team members will engage in user interface development, Christa will be at the forefront of this task, focusing on user engagement and functionality. This phase will serve as a learning opportunity for the team to understand system architecture intricacies. Rishab will take charge of indexing and integrating the algorithm. Their role involves applying the BM25 algorithm and developing the indexing structure. Stef will lead the testing and query refinement, evaluating the system against established benchmarks to enhance precision and recall. This process will refine our search algorithm. All team members are involved in quality assurance and documentation, maintaining high standards and detailed records of the project's development for clarity and future reference. Regular progress meetings are scheduled to align the team on progress and future tasks, helping facilitate informed decision-making. The final stages involve collective effort in evaluating and optimising the search engine, with each member providing feedback to refine its performance. Our final report and presentation will be prepared together, reflecting the process of our project from its inception to its completion.

Time Plan

The development of our search engine is scheduled over five weeks starting from 5th March 2024 to April 10th 2024. The project begins with data collection on March 5th and is expected to last one week. The document cleaning and preprocessing will start on March 8th. These actions will run alongside the first progress meeting on March 13th to assess initial progress and prepare for the system design phase.

The system design phase is scheduled to start on March 12th, overlapping with preprocessing, allowing design and data preparation. This phase will continue for six days, concluding with a progress meeting on March 19th. This will allow for a transition to indexing and BM25 algorithm integration, starting from March 19th to March 25th, followed by a prototype completion milestone on March 26th. Following the prototype review, we will finalise the system architecture by March 31st. Alongside the finalisation phase, the user experience and interface design phase will begin on March 20th and run until April 2nd, allowing for the development of interface concepts based on the system's capabilities.

As shown in *Figure 2*, the final phases, including functionality testing and query refinement, are scheduled from April 1st and include a comprehensive system evaluation on April 6th. These stages are fundamental to ensure the system's effectiveness and user satisfaction. Throughout the project, documentation will be developed in parallel to capture the process and findings, which will be completed on April 6th. A final progress meeting on April 7th will confirm the project's readiness for the final presentation. The project concludes with the complete evaluation and presentation drafted on April 10th, ensuring all phases are thoroughly documented and results are effectively communicated. This timeline ensures that all project objectives are met within the timeframe.



PROJECT: Machine Learning Journals - Search Engine

QMUL - Information Retrieval
ECS736U/ECS736P
Project start date: 05/03/2024

Legend: On track | Low risk | Med risk | High risk

Milestone description	Category	Assigned to	Progress	Start	Days
Data Management:					
Dataset Collection	Low Risk	All	5%	05/03/2024	9
Dataset Preprocessing	Med Risk	Stef	0%	08/03/2024	7
Progress Meeting	Low Risk			13/03/2024	1
Data Preprocessing Completion	Milestone			15/03/2024	1
System Architecture:					
System Design	High Risk	Christa	0%	12/03/2024	6
Progress Meeting	On Track			19/03/2024	1
Indexing and Algorithm Integration	Med Risk	Rishab	0%	19/03/2024	6
Prototype Complete	Milestone			26/03/2024	1
Progress Meeting	Low Risk			27/03/2024	1
Finalise Architecture	On Track			27/03/2024	4
Interface:					
User Experience Design	Med Risk	All	0%	20/03/2024	6
User Interface Development	High Risk	All	0%	27/03/2024	6
Quality Assurance:					
Functionality Testing and Query Refinement	Med Risk	Stef	0%	01/04/2024	4
Documentation	High Risk	All	0%	03/04/2024	3
Comprehensive Evaluation	Med Risk		0%	06/04/2024	1
Project Debrief	Milestone			07/04/2024	1
Presentation Preparation	Low Risk		0%	07/04/2024	3
Post-Evaluation Adjustments	Med Risk	All	0%	08/04/2024	2
Finalisation of project	Milestone			10/04/2024	1

March
April
T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

Figure 2: Gantt Chart of Project Schedule for the Development of Search Engine

References

Boeker, M., Vach, W. and Motschall, E. (2013) *Google scholar as replacement for systematic literature searches: Good relative recall and precision are not enough - BMC Medical Research methodology*, *BioMed Central*. Available at: https://doi.org/10.1186/1471-2288-13-131 (Accessed: 02 March 2024).

Chaput, M. (2012) Introduction to whoosh¶, Introduction to Whoosh - Whoosh 2.7.4 documentation. Available at: https://whoosh.readthedocs.io/en/latest/intro.html (Accessed: 07 March 2024).

Gusenbauer, M. and Haddaway, N.R. (2020) *Which academic search systems are suitable for systematic reviews or meta-analyses? evaluating retrieval qualities of google scholar, pubmed, and 26 other resources*, *Research synthesis methods*. Available at: https://pubmed.ncbi.nlm.nih.gov/31614060/ (Accessed: 03 March 2024).

Hajba, G.L. (2018) 'Using Beautiful soup', Website Scraping with Python, pp. 41–96. doi:10.1007/978-1-4842-3925-4_3.

Hardeniya, N. et al. (2016) Natural language processing: Python and NLTK: Learn to build expert NLP and machine learning projects using NLTK and other Python Libraries: A course in three modules. Birmingham ; Mumbai: Packt.

Loeliger, J. and McCullough, M. (2012) Version control with git: Powerful tools and techniques for collaborative software development ; covers GitHub. Beijing: O'Reilly.

Manning, C.D., Schütze, H. and Raghavan, P. (2008) 'Introduction to Information Retrieval', *Information Retrieval*, 12(5), pp. 609–612. doi:10.1007/s10791-009-9096-x.

Miller, C. (2018) Hands-on data analysis with NumPy and pandas: Implement python packages from Data Manipulation to processing. Birmingham: Packt Publishing Ltd.

Rajaraman, A. and Ullman, J.D. (2011) *Mining of massive datasets*, *Cambridge Core*. Available at: https://doi.org/10.1017/CBO9781139058452 (Accessed: 03 March 2024).

Tiangolo (2018) FASTAPI, FastAPI. Available at: https://fastapi.tiangolo.com/ (Accessed: 07 March 2024).