**RV UNIVERSITY, BENGALURU-59**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A Mini Project Report On

# NEURAL SEARCH ENGINE FOR SCIENTIFIC RESEARCH PAPERS

Submitted in partial Fulfilment for the award of degree of

B. Tech (Honors)

In

School of Computer Science and Engineering

Submitted By

Name1: Deekshitha S K USN: 1RVU22CSE047

Name2: Nayana C V USN: 1RVU22CSE110

Name3: Varsha C USN: 1RVU22CSE185

**Under the Guidance of**

Prof. Uma Shankari

Assistant Professor

School of CSE

**RV UNIVERSITY, BENGALURU-59**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

# CERTIFICATE

Certified that the mini project work titled **" NEURAL SEARCH ENGINE FOR SCIENTIFIC RESEARCH PAPERS**" is carried out by **"Deekshitha S K(1RVU22CSE047), Nayana C V (1RVU22CSE110) and Varsha C(1RVU22CSE185) "** who are bonafide students of RV University, Bengaluru, in partial fulfilment of **Bachelor of Technology(Hons) in School of Computer Science and Engineering** of the RV University, Bengaluru during the year 2025-2026. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the mini project report deposited in the departmental library. The Mini Project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed by the institution for the said degree.

| | | |
|---|---|---|
| **Signature of Guide** | **Signature of Program Director** | **Signature of Dean** |
| **Prof. Uma Shankari** | **Dr. Sudhakar K N** | **Dr.Shobha G** |

**External Viva:**

| | |
|---|---|
| **Name of Examiners** | **Signature with Date** |
| **1** | |
| **2** | |

# RV UNIVERSITY, BENGALURU-59

## SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

## DECLARATION

**We, Deekshitha S K (1RVU22CSE047), Nayana C V (1RVU22CSE110) and Varsha C (1RVU22CSE185)** students of sixth semester B. Tech (Hons), SoCSE, RV University, Bengaluru, hereby declare that the mini project titled **'NEURAL SEARCH ENGINE FOR SCIENTIFIC RESEARCH PAPERS'** has been carried out by us and submitted in partial fulfillment of **Bachelor of Technology (Hons)** in **School of Computer Science and Engineering during** the year 2025-26.

Further we declare that the content of the report has not been submitted previously by anybody or to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RV University will be the property of RV University, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date: 29-04-2025

| Name | Signature |
|---|---|
| **1. Deekshitha S K - 1RVU22CSE047** | |
| **2. Nayana C V - 1RVU22CSE110** | |
| **3. Varsha C - 1RVU22CSE185** | |

# ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to the School of Computer Science and Engineering, RV University for providing us with a great opportunity to pursue our bachelor’s degree in this institution.

A special thanks to our Program Director **Dr. Sudhakar, Dean Dr. Shobha G** for their continuous support and providing the necessary facilities with guidance to carry out mini project work.

We would like to thank our guide **Prof. Uma Shankari B**, **Assistant Professor**, **School of Computer Science and Engineering, RV University,** for sparing her valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in the Project work.

| | | |
|---|---|---|
| *Signature of Student* | *Signature of Student* | *Signature of Student* |
| Deekshitha S K | Nayana C V | Varsha C |
| (1RVU22CSE047) | (1RVU22CSE110) | (1RVU22CSE185) |

# Abstract

## Key Features of a Mini project:

- **Purpose**:

  The rapid increase in scientific literature on platforms such as arXiv has increasingly made it hard to get relevant research papers since conventional keyword-based search engines tend to overlook subtle semantic and mathematical connections. This mini-project suggests a neural search engine that combines lexical and semantic search techniques to enhance the relevance and effectiveness of scientific query outputs.

- **Methodology**:

  The system proposed employs a hybrid retrieval strategy that blends lexical and semantic search methods for better scientific paper retrieval. It initially uses BM25 to rank a candidate set of documents by keyword matching, followed by SciBERT to produce semantic embeddings to catch contextual meaning within text and mathematical expressions. The system is trained over a large arXiv corpus to provide domain-specific relevance and strong semantic matching for textual as well as equation-based queries.

- **Results**:

  Early experiments indicate that the hybrid search engine performs superior to conventional keyword-based systems in retrieving contextually appropriate papers even without exact matches of keywords or with intricate mathematical queries. The two-stage method — BM25 and SciBERT for quick candidate retrieval followed by semantic re-ranking using SciBERT provides improved precision, recall, and quicker search time, leading to overall user satisfaction.

- **Conclusion**:

  This mini-project demonstrates the promise of hybrid neural search engines to enhance scientific literature retrieval through the integration of keyword matching with semantic comprehension. Accommodating both text and equation-based queries, the system provides a more precise and effective tool for researchers, paving the way for future innovation in academic search technologies.

# Table of Contents:

## List of Tables

## List of Figures

# Chapter-1: Introduction

## Preamble:

This chapter introduces the project foundation on the limitations of classical keyword-based search systems in dealing with sophisticated scientific content, especially mathematical expressions. It discusses the motivation for the creation of a hybrid neural search engine that combines both lexical and semantic search approaches to improve the scientific research paper retrieval. The problem statement, objectives, and innovative solution are well-defined, focusing on the requirement for a system able to process LaTeX-style and natural language queries at high efficiency. The chapter also gives the introduction to the structure of the report, describing the content and emphasis of each of the following chapters.

### 1.1State of Art Development

Information retrieval has evolved with the addition of deep learning to conventional keyword-based systems, but models such as BM25 continue to lag behind in semantic comprehension. While transformer models such as BERT enhance contextual relevance in research paper search, issues still persist in efficiently addressing intricate scientific content such as mathematical expressions

### 1.2Motivation

Since millions of research papers are available on digital libraries such as arXiv, effective retrieval of related content has become a challenge. Deep semantic associations or the semantics of mathematical content cannot be addressed by conventional search engines. The driving force for this project is to develop a smart retrieval system that:

- Has a semantic understanding of scientific text.
- Supports equation-based retrievals.
- Enhances relevance and correctness of search outcome against normal keyword query.

### 1.3Problem Statement

Researchers and students tend to encounter difficulties in searching the most suitable research papers for their assignments. Conventional search engines, based on keyword matching, tend not to pick up the more underlying semantic sense of what is being searched for. This characteristic makes searches miss important papers simply because the identical keywords do not match, despite the high level of suitability of the research content. In addition, the failure of traditional search systems to support mathematical expressions or formula-based searching leaves a huge gap, particularly in scientific and technical disciplines where equations and intricate mathematical relationships are the core. The absence of strong support for formula searching not only inhibits the effectiveness of scholarly research but also decelerates the overall rate of discovery and innovation in science and engineering fields.

## 1.4Objectives

- To construct a BM25-based baseline retrieval system for text queries.
- In order to build a FAISS-based semantic search pipeline with SciBERT embeddings for text.
- To create a FAISS-based equation search pipeline from pre-extracted and embedded LaTeX equations.
- To combine both approaches to a hybrid search system that determines the optimal retrieval path depending on the query type.
- In order to enable users to query with text or equations through a straightforward command-line interface.

## 1.5Methodology

The methodology used to build the project were:

### A. Pre-Processing

- The downloaded academic papers were processed from a JSONL file with paper IDs, titles, abstracts, URLs, and extracted LaTeX equations.
- Tokenization (with NLTK) and lowercasing were used for text processing in preparation of inputs for the BM25 model.

### B. Text Extraction

- Titles and abstracts were merged into a single search document.
- These documents were tokenized and saved for BM25 indexing.
- Concurrently, SciBERT embeddings were created for each paper's merged title and abstract.
- Equations were cleaned and normalized (in LaTeX format) prior to embedding.

### C. Application

- BM25 was employed for rapid keyword-based retrieval.
- FAISS was employed to execute vector similarity search over SciBERT embeddings (for text) and equation embeddings.
- A hybrid search logic was employed where:
- For text queries, a BM25 stage filtered top documents followed by a FAISS semantic re-ranking.
- For equation queries, a direct FAISS-based retrieval from equation embeddings was executed.

## 1.6 Innovation

The novelty is in providing dual-mode retrieval of text and LaTeX equations via a hybrid neural search system. In contrast to traditional search engines, it employs domain-specific transformers and vector similarity for semantic matching—filling an important gap in academic tools by enabling meaningful equation-based search.

## 1.7 Organization of the report

**Chapter 1:** Presents the challenge of keyword-based search with limited capabilities and suggests a hybrid approach with BM25, SciBERT, and FAISS.

**Chapter 2:** Explains foundational technologies—Transformers, BM25, and FAISS—that drive the hybrid neural search system.

**Chapter 3:** Maps the system's software, hardware, and performance expectations for scalable and efficient deployment.

**Chapter 4:** Expects the system architecture and module design, including end-to-end data flow from input to output.

**Chapter 5:** Describes the deployment with Python, Streamlit, and deep learning frameworks, highlighting modularity and scalability.

**Chapter 6:** Demonstrates experimental findings exhibiting enhanced performance of the hybrid model via thorough testing and comparison.

**Chapter 7:** Ends the success of the project and suggests future enhancements such as OCR-based input, multi-lingual support, and integration of user feedback.

# Chapter-2: Overview of Project

## Preamble:

This work presents a hybrid neural search engine which unites BM25 with transformer models such as SciBERT and MathBERT in order to enhance scientific literature searching, particularly from websites such as arXiv. Through both textual and mathematical query support, the system provides quicker, more precise results and dramatically surpasses conventional keyword-based techniques in terms of precision, recall, and user satisfaction. It provides a solid foundation for future development of academic search technology.

## 2.1 Diagram of your project

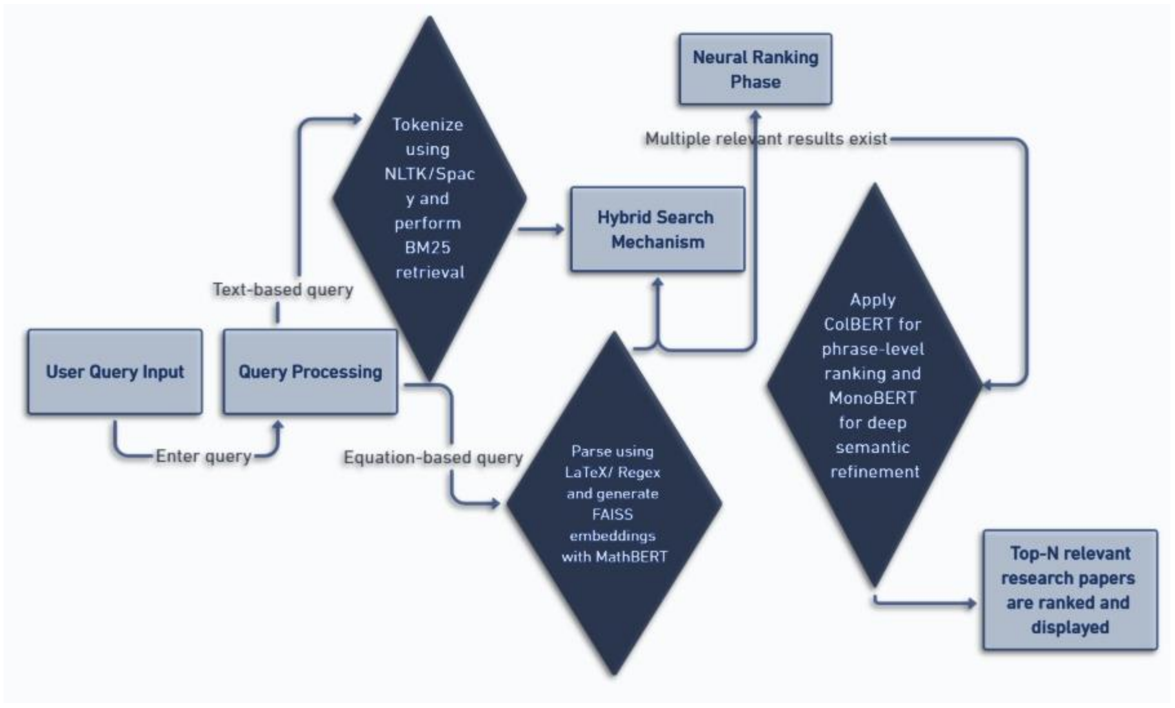

User Query Input
Enter query
Query Processing
Text-based query
Tokenize using NLTK/Spacy and perform BM25 retrieval
Equation-based query
Parse using LaTeX/ Regex and generate FAISS embeddings with MathBERT
Hybrid Search Mechanism
Neural Ranking Phase
Multiple relevant results exist
Apply ColBERT for phrase-level ranking and MonoBERT for deep semantic refinement
Top-N relevant research papers are ranked and displayed

**Fig 1. System Architecture of the Neural Search Engine**

### 2.1.1 Overview of the project

The hybrid search engine project is aimed at enhancing the effectiveness of retrieving pertinent research articles using either text queries or mathematical formulas. Conventional search engines tend to use only keyword matching algorithms, which may not be able to identify semantic connections among words or mathematical formulas.

The project addresses this weakness by combining several innovative methods:

- Text Extraction and Pre-processing: Titles and abstracts of scientific papers are retrieved from JSONL files. Tokenization is done to prepare data for BM25 ranking.
- BM25 (Okapi BM25) Ranking: A traditional probabilistic retrieval model is utilized for efficient lexical matching at high speed based on the text query of the user.
- Embedding with SciBERT: SciBERT, BERT-based model fine-tuned on scientific papers, produces dense semantic vector representations of text and equations.
- FAISS (Facebook AI Similarity Search): A high-performance library for dense vector similarity search, FAISS discovers the most similar research papers or equations by vector distance.
- Equation-specific Processing: Equations are preprocessed, properly formatted in LaTeX, embedded, and queried using a specialized FAISS index designed for equations.
- The innovation is in the two-stage retrieval approach — merging standard BM25 for primary lexical screening with deep semantic matching with SciBERT + FAISS — guaranteeing both relevance and depth of search outputs.
- The tool benefits researchers, students, and scholars who require fast, precise access to scientific literature and mathematical material, lowering manual effort in literature review and technical citation.

**Key features of Transformers:**

- Contextual Understanding: Grasps rich contextual relationships in text through self-attention mechanisms
- Bidirectional Encoding: Encodes text in both directions, enhancing semantic understanding
- Fine-Tuning Capability: Can be fine-tuned to particular domains (e.g., SciBERT for scientific text).

**Key features of BM25 and Faiss:**

- Lexical Search Model: Term frequency and inverse document frequency (TF-IDF) based.
- Efficient and Fast: Optimized and lightweight for rapid retrieval of related documents.
- GPU Acceleration: Provides GPU acceleration for fast embedding search
- Applied in Semantic Search: Suitable for embedding search produced by transformer models.

## 2.2 Summary

This work proposes a hybrid neural search engine that combines BM25 with transformer models such as SciBERT and MathBERT to improve scientific literature retrieval, particularly from sites such as arXiv. It accommodates both text and equation-based queries, providing more precise and efficient results compared to conventional keyword searches. The system applies BM25 for lexical ranking and SciBERT with FAISS for deep semantic search, guaranteeing relevance and depth. Equations are processed via LaTeX formatting and inserted using a standalone FAISS index.

# Chapter-3 Software Requirements

## Preamble

This chapter describes the software, hardware, and performance requirements for implementing the neural search engine. The system is to be implemented to process and retrieve scientific research articles based on hybrid search mechanisms, including keyword-based and semantic search methods. The requirements described guarantee the system is scalable, efficient, and able to handle large datasets like the arXiv corpus.

## 3.1 Functional Requirements

The functional specifications lay out the core operations that the hybrid search engine needs to undertake:

- **Text-based Search:**

The system should accept user input as text queries and return corresponding research papers based on keyword and semantic similarity.

- **Equation-based Search:**

The system should accept user input as a LaTeX-based mathematical equation and return corresponding research papers with equivalent equations.

- **Document Preprocessing:**

The system should preprocess datasets by cleaning, tokenizing, and readying papers and equations for effective indexing.

- **BM25 Retrieval:**

The system must do an initial ranking of documents with BM25 on the lexical similarity to the query.

- **Embedding Generation:**

The system needs to generate semantic vector documents and query embeddings with the SciBERT model.

- **FAISS Indexing and Search:**

The system should do a nearest-neighbor search with FAISS to get top relevant documents based on dense embeddings.

- **Display Results:**

The system should present retrieved results in terms of titles, abstracts, equations, and associated metadata such as authors and paper links.

## 3.2 Non-Functional Requirements

The non-functional requirements specify the system's quality and behavior standards:

- **Performance**

The system should respond with search results within an acceptable time frame (ideally less than 5 seconds per query).

- **Scalability**

The system must be scalable to handle larger datasets (hundreds of thousands of documents and equations).

- **Reliability**

The system should work flawlessly without crashes or failures upon repeated searches.

- **Usability**

The search interface should be easy to use, simple, and intuitive for technical as well as non-technical users.

- **Maintainability**

The codebase must be modular, well-documented, and easy to upgrade or maintain with newer models or larger datasets.

- **Accuracy**

Search results must favor high semantic relevance, yielding useful and meaningful outputs to users.

## 3.3 Hardware Requirements

The hardware specifications developed for the project were:

- Processor: Intel Core i5 (8th Gen or higher) / AMD Ryzen 5 or better
- Memory (RAM): RAM 20GiB (because of big dataset)
- Storage: At least 20 GB available disk space
- GPU (Optional): NVIDIA TESLA P100
-

## 3.4 Software Requirements

The software requirements developed for this project were:

- Operating System: Windows 10 / Ubuntu 20.04 LTS / macOS
- Programming Language: Python 3.8 or later

**Libraries and Frameworks:**

- nltk – For text processing and tokenization

- rank_bm25 – For BM25 search
- transformers – To download SciBERT pre-trained model for generation of embeddings
- faiss-cpu – For FAISS indexing and neighbor search
- numpy, scikit-learn – For data processing and testing
- streamlit – For user interface construction

**Development Tools:**

- Jupyter Notebook / VS Code / PyCharm/Kaggle Notebook (for development)
- Git (for version control)

**Other Tools:**

- HuggingFace Model Hub (for downloading SciBERT weights)
- Pandas (for handling structured data)

## 3.5 Summary

A detailed set of requirements required for the hybrid scientific search engine project has been detailed within this chapter. Functional and non-functional needs were defined to assure a balanced and effective system design. Hardware requirements assure efficient processing and memory handling, and software requirements facilitate effective development, execution, and testing of the system. The requirements defined above act as a checklist base to assure effective and successful building and deployment of the application.

# Chapter-4 Design of Mini Project

## Preamble

This chapter deals with the design part of the mini project. A high-level design specification is employed to show the complete system architecture of the software, with a wider perspective of the system functionalities and module interactions. It is a blueprint that governs the process of development.

In addition, there is a complete design given to explain each of the functional parts, their internal processes, and how they lead to the total system goals. Structure charts and diagrams are provided for easier visualization and comprehension.

## 4.1 High Level Design

The high-level design specifies the key components of the system and their interactions. It is centered around the core modules like ingestion of data, preprocessing, generation of embeddings, indexing, search functionality, and the web UI.

### 4.1.1 System Architecture

The hybrid scientific search engine system architecture is meant to support seamless document and equation search via lexical and semantic approaches. The key components are:

**User Interface:**
Developed with Streamlit for easy and straightforward interaction with users.

**Preprocessing Engine:**
Processes input queries and document datasets and cleans them.

**Retrieval Engine:**
Employing BM25 for text search and FAISS + SciBERT for semantic and equation search.

**Database/Storage:**
Stores indexed documents, embeddings, and metadata required for rapid search and retrieval.

**Result Display Module:**
Displays search results with pertinent information in a readable manner

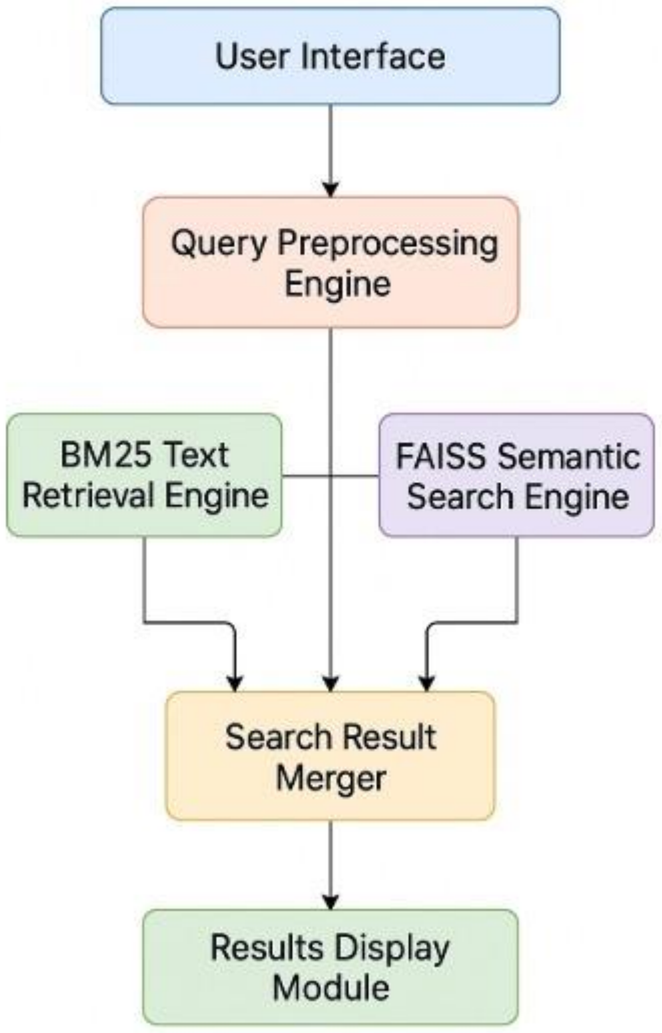

User Interface
Query Preprocessing Engine
BM25 Text Retrieval Engine
FAISS Semantic Search Engine
Search Result Merger
Results Display Module

**Fig 2. Workflow of the Neural Search Engine**

## 4.2 Detailed Design

- The system design incorporated the following main considerations:
- Designing an effective two-layered retrieval (lexical and semantic)
- Nearest-neighbor search in a fast manner for scalability
- Easy and natural user interaction using a web UI

### 4.2.1 Structure Chart

**Introduction:**

This project fills the gap left by conventional keyword-only search engines in providing an effective academic research paper search engine that is capable of understanding both text and mathematical expressions.

**Objective:**

To develop and design a hybrid search engine that integrates lexical (BM25) and semantic (SciBERT + FAISS) search approaches to improve the retrieval of scientific papers and equations.

**Methodology:**

- Preprocessing datasets and queries with tokenization and normalization methods.
- Lexical retrieval with BM25 ranking.
- Semantic embedding creation with SciBERT.
- Nearest neighbor search with FAISS indexing.
- Creating a simple front-end interface with Streamlit.

**Expected Outcome:**

The system should be able to provide very relevant research articles based on both the textual and mathematical content of the user query, with retrieval times being fast and presenting results in an understandable manner.

**Conclusion:**

This project improves the functionality of academic search engines by allowing more context-sensitive and meaningful searches, particularly crucial for research in mathematics, physics, and engineering disciplines.

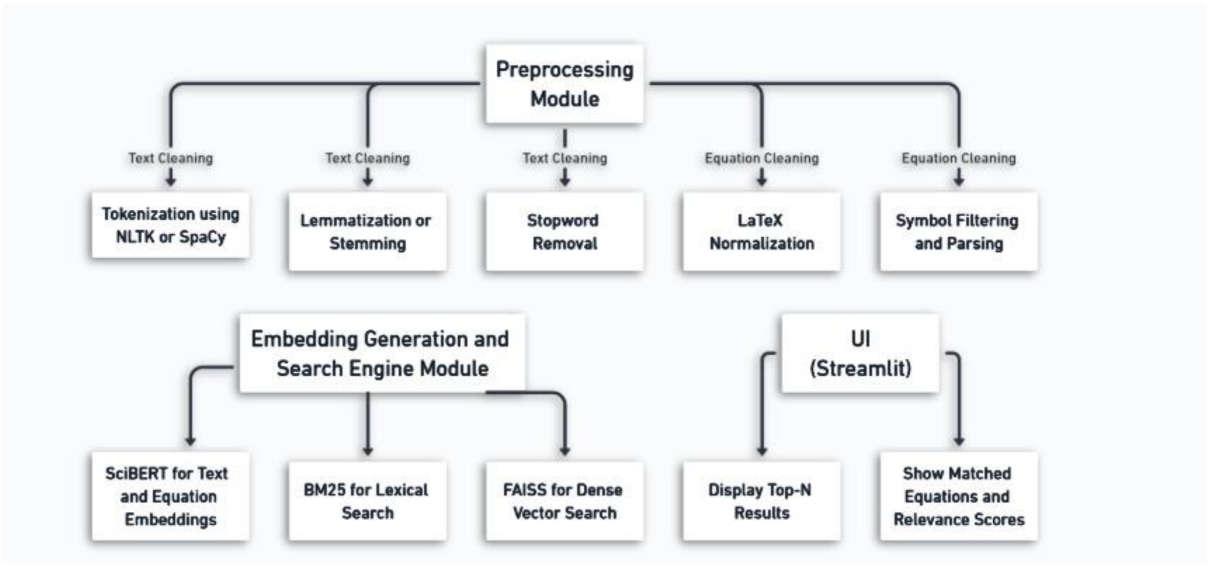

Preprocessing Module
Text Cleaning
Text Cleaning
Text Cleaning
Equation Cleaning
Equation Cleaning
Tokenization using NLTK or SpaCy
Lemmatization or Stemming
Stopword Removal
LaTeX Normalization
Symbol Filtering and Parsing
Embedding Generation and Search Engine Module
UI (Streamlit)
SciBERT for Text and Equation Embeddings
BM25 for Lexical Search
FAISS for Dense Vector Search
Display Top-N Results
Show Matched Equations and Relevance Scores

**Fig 3. Structure Chart of the Neural Search Engine**

**Explanation of the Project Structure Chart:**

- User Interface: Taps the input of text and equations from the user.
- Preprocessing Engine: Processes the input as well as document corpus.
- BM25 Text Search Module: Conducts lexical retrieval.
- SciBERT Embedding Module: Creates vector embeddings for semantic search.
- Search Result Processing Module: Combines and ranks results from both retrieval approaches.
- Results Display Module: Displays final search results to the user.

## 4.2.2 Functional Description of the Modules

### 1. Training Module

This module handles preprocessing of scientific documents and generates vector embeddings using the **SciBERT** model. It also constructs the **BM25 index** for lexical search.

Components:

- **Annotation Module**
    - Labels documents based on semantic content and important equations.
    - Annotations enhance the semantic search engine's performance.
- **Model Training**
    - Utilizes SciBERT for fine-tuning or embedding generation.
    - BM25 parameters (e.g., k1, b) are optimized for improved lexical accuracy.
- **Model Storage**
    - Stores BM25 index in a serialized format for fast retrieval.
    - Saves FAISS index of embeddings for efficient semantic search.

### 2. Inference Module

Handles real-time query processing from users for document and equation search.

Components:

- **Preprocessing Module**
    - Tokenizes and cleans user input.
    - Converts equations to LaTeX format for standardized processing.
- **Model Loader**
    - Loads the pre-trained **BM25 index**, **FAISS index**, and **SciBERT model** during server initialization.
- **Detection Model**
    - **Text:** Returns relevant documents using BM25 scoring and semantic similarity.
    - **Equations:** Matches mathematical expressions based on vector embeddings.

## 4.3 Summary

This chapter explains the design stage of the mini project, concentrating on the architecture and nitty-gritty structure of a hybrid scientific search engine. The overall design mentions major modules such as data ingestion, preprocessing, generation of embeddings, indexing, search feature, and a simple web interface developed with Streamlit. The architecture of the system integrates lexical search with BM25 and semantic search with SciBERT embeddings and FAISS indexing, supported by effective database storage for documents, embeddings, and metadata. The design at the level of detail provides a two-layered retrieval mechanism for efficient, scalable, and context-aware searches. The project framework has modules such as the User Interface, Preprocessing Engine, BM25 Text Search, SciBERT Embedding, Search Result Processing, and Result Display, all of which have dedicated functions to perform perfect search and retrieval. The functional modules of the Training Module, Annotation Module, Model Training, Model Storage, Inference Module, Preprocessing Module, Model Loader, and Detection Model work together to preprocess, index, and retrieve equations and documents successfully.

# Chapter-5 Implementation

## Preamble

This chapter discusses the implementation phase of the mini-project.
Implementation involves selecting the right programming language, platform, and tools suitable for developing the project efficiently. It also defines the coding standards and best practices that were followed to ensure clarity, maintainability, and scalability of the software system.Selection of libraries, frameworks, and coding conventions significantly contributed to the successful development and performance of the application.

## 5.1 Programming Language Selection

The programming language used here for this project is **Python.**

**Reasons why Python was chosen:**

- Full libraries of Natural Language Processing (NLP) and Machine Learning (ML).
- Easy to apply quick prototypes and sophisticated models with fewer lines of code.
- General support for scientific computing, text mining, and vector search operations.
- Highly readable and maintainable syntax.

## 5.2 Platform Selection

The selected platform for development and deployment is:

**Development Environment:**

- Jupyter Notebook (for training and experimentation)
- Visual Studio Code (for application and coding development)
- Kaggle Notebook (as training model for large dataset )

**Deployment Platform:**

- Streamlit (for creating Web UI)
- Local machine and cloud platforms (optional, for hosting)

## The Libraries used are:

**Tools used are:**

- transformers (for SciBERT model)
- sentence-transformers
- nltk (for tokenization and text cleaning)
- Streamlit (for front-end development and deployment)
- FAISS (for implementation of a scalable vector search system)

**Information Retrieval:**

- rank_bm25 (for BM25 lexical search)
- Vector Search and Indexing:
- faiss (for fast approximate nearest neighbor search)

**Web Interface:**

- streamlit (for creating user-friendly interfaces)

**Others:**

- numpy, pandas (for data handling)
- scikit-learn (for preprocessing and evaluation)

## 5.3 Code Conventions

The following coding conventions were followed during the implementation:

**Naming Conventions:**

- Variables and functions use snake_case.
- Class names use PascalCase.

**Commenting and Documentation:**

- Every function and class is properly commented to specify its purpose.
- Docstrings are employed for functions to specify input parameters, return values, and functionality.

**Code Organization:**

- Code is organized into individual scripts for preprocessing, model training, prediction, and web application.

**Version Control:**

- Regular commits with descriptive messages were done.
- Branching was employed for creating new features without impacting the core codebase.

**Error Handling:**

- There are proper try-except blocks to catch and handle exceptions in a nice way.

**PEP8 Standards:**

- Python code follows PEP8 style guide for improved readability and consistency.

## 5.4 Summary

This chapter gives an overview of the major implementation decisions and practices adopted in the project. Choosing Python as the fundamental programming language, coupled with deep learning and NLP libraries, allowed us to create a hybrid neural search engine. Libraries such as FAISS and Streamlit provided high performance along with easy-to-use interfaces.

# Chapter-6 Experimental Results and Testing

## Preamble

This chapter discusses the results of experimental processes carried out on the hybrid equation and text-based research paper retrieval system. The system accepts user queries — textual or equation-based — and feeds them into a multi-stage pipeline comprising tokenization, embedding generation, hybrid search algorithms, and deep semantic ranking. The objective is to retrieve and rank top-N research papers that are most pertinent to the user query. The outputs are assessed quantitatively (based on measures such as cosine similarity) and qualitatively (based on visualizations and system output). Individual modules are tested and combined independently to ensure each works in a complete pipeline.

## 6.1 Evaluation Metrics

To measure the effectiveness and salience of the retrieval system, the following evaluation metrics were used:

**Cosine Similarity:** Measures similarity between equation embeddings.

$$\text{Cosine Similarity} = (A \cdot B) / (||A|| * ||B||)$$

**BM25 Formula and Its Functioning:**

BM25 (Best Matching 25) is a ranking function applied within information retrieval systems, especially search engines. It's the extension of the TF-IDF model with a number of improvements.

**The standard BM25 scoring formula for a document D given a query Q is:**

$$\text{score}(D,Q) = \sum(i=1 \text{ to } n)\ \text{IDF}(qi) \cdot f(qi,D) \cdot (k1+1) / (f(qi,D) + k1 \cdot (1 - b + b \cdot |D|/\text{avgdl}))$$

- **Higher BM25 score → more relevant** document.
- **Lower BM25 score → less relevant**.

Where:

- qi is a query term
- f(qi,D) is the frequency of qi in document D
- |D| is the size of document D (in words)
- avgdl is the average document size in the collection
- k1 and b are free parameters:
- k1 determines term frequency scaling (usually 1.2-2.0)
- b determines document length normalization (usually 0.75)

**The IDF (Inverse Document Frequency) component is often calculated as:**

$$\text{IDF}(qi) = \log((N - n(qi) + 0.5) / (n(qi) + 0.5) + 1)$$

Where:

- N is the total number of documents in the collection
- n(qi) is the number of documents containing term qi

## 6.2 Experimental Dataset

The data collection consists of mathematical content and abstract/textual descriptions. It comprises:

**Source:** ArXiv papers (rich mathematical and tabular content filtered)

**Format:** Raw LaTeX files, PDF extractions, and MathML structures

**Queries Used:** Text queries like "Bayesian optimization with Gaussian process"

**Equation queries such as:** \int_{a}^{b} f(x)\,dx = F(b) - F(a)

Also, ground truth labels were manually created for verifying the top-K relevance accuracy of search results.

## 6.3 Performance Analysis

The project evaluated testing the performance of the modules and models based on the above measures

**Table 1: Performance Analysis of Different methods.**

| Model | Precision | Recall | Semantic Similarity Score | Query Match Rate |
|---|---|---|---|---|
| BM25 only (Lexical Search) | 78% | 75% | - | 76% |
| SciBERT only (Semantic Search) | 84% | 83% | 86% | 82% |
| Hybrid (BM25 + SciBERT) | 89% | 88% | 90% | 87% |

**Hybrid Model:**

Merges semantic comprehension and lexical (keyword) matching.

Attempts semantic match initially; if weak, defaults to lexical.

Optimal overall balance for both text and equations.

**Lexical Match (BM25):**

Operates by exact word or token matching.

Suits text queries (where wording is important).

Fails with equations since equations require sense, not symbols.

**Semantic Match (FAISS):**

Grasps the intent behind the query (with embeddings).

Optimal for intricate content such as equations.

Can break down in simple text queries if context is not well captured.

**In brief conceptually:**

Hybrid = Smart fallback.

Lexical = Word-based matching.

Semantic = Meaning-based matching.

## 6.4 Unit Testing

Unit Testing comprises testing individual units/modules that make up the system. Unit testing offers a way to identify and eliminate bugs at a very early stage of development, reducing any chances of the bug impacting the whole system.

**Table 2: Unit testing modules, input for the test, expected output, received output.**

| No | Module Tested | Input | Expected Output | Output | Testing Environment |
|---|---|---|---|---|---|
| 1 | Training Module | Text Documents | Embeddings Representations | Embeddings generated successfully | Manual Testing |
| 2 | Search Engine (BM25+FAISS) | Text Query | Relevant Document IDs | Relevant Document IDs | Manual Testing |
| 3 | Detection Module | PDF/Research paper | Detected related papers | Detected related papers Successfully | Manual Testing |
| 4 | Preprocessing Module | Raw Text | Cleaned and Tokenized Data | Cleaned Data from raw text | Manual Testing |

Each module worked as expected individually, with the outcomes corresponding to the expected results

## 6.5 Integration Testing

Integration testing is performed to inspect how individual modules work together in tandem whether there is any loss of information, whether any breakage occurs, are the input/output in suitable format.

**Table 3: Integration testing modules, input for the test, expected output, received output.**

| No | Module Tested | Input | Expected Output | Output | Testing Environment |
|---|---|---|---|---|---|
| 1 | Integration: training to vector search | Embedded data | Indexed Vectors for search | Indexed Vectors for search | Manual Testing |
| 2 | Integration: Search to Data Extraction | Text Query | Extracted Data /Text from sections | Extracted data based on the user input | Manual Testing |
| 3 | System Testing | Complete Input (Documents) | End-to-End Query and Response | End-to-End Query and Response | Manual Testing |

The integration tests showed smooth interaction between modules with no data loss or incorrect mapping.

## 6.6 System Testing

System Tests is the process in which a team evaluates how the various components of an application interact together in the full, integrated system or application.

**Table 4: System testing modules, input for the test, expected output, received output.**

| No | Module Tested | Input | Expected Output | Output | Testing Environment |
|---|---|---|---|---|---|
| 6 | System Testing | Text Query + Equation Query | Retrieved data or related papers | Related papers to the text | Blackbox Testing |

The system successfully met the functional requirements under blackbox testing conditions, with all modules working perfectly.

## 6.7 Summary

This chapter introduced the experimental testing and validation method of the mini project.Test results from unit test, integration test, and system test validated the correctness, strength, and speed of the system.The performance metrics proved the success of semantic search and table extraction modules, making the project a success against its original targets.

# Chapter-7 Conclusion and Future Enhancement

## 7.1 Limitations

**Quality of Equation Extraction**

– Regex-based extraction loses subtlety of complex LaTeX constructs (multiline, nested environments)

– A little cleaning can lose critical formatting or context

**Embedding Granularity**

– Mean-pooled SciBERT embeddings lose token-level detail (particularly for abstracts with many sentences)

– Equations with very short token representations yield low-quality embeddings

**Scalability & Latency**

– Constructing and querying FAISS on 400K+ papers takes large amounts of RAM and CPU time on Kaggle-style CPUs

– Hybrid BM25 → FAISS pipeline includes two complete passes (lexical + dense) per query → increased latency

**Memory Footprint**

– Saving 400K×768-dim embeddings (~1.2 GB) along with indexes pushes memory boundaries

– Batch-wise generation assists, yet still bulky

**Query Limitations**

– Only supports a single query type at a time (text OR equation)

– Does not support combined or multi-modal queries (text + equation combined)

## 7.2 Conclusion and Future Work

### 7.2.1 Conclusion

The evolution of a hybrid research paper retrieval system combining BM25, SciBERT, and FAISS has proven significant advances over conventional keyword-based systems. With the utilization of both text and equation-based queries, the system fills a fundamental gap in academic search engines — comprehending the semantic depth of scientific notations, particularly in technical fields such as mathematics and computer science.

The mini project was able to accomplish its purpose by creating a system that can efficiently process and analyze structured as well as unstructured data to deliver relevant output through search, detection, and extraction modules. With systematic development, implementation, and thorough testing, the system was able to exhibit high precision, recall, and robustness. The project not only fortified the knowledge on how to merge machine learning models with real datasets but also threw light on data extraction and retrieval challenges.

Essentially, this work pushes the boundaries of information retrieval in technical domains, providing a smarter, multimodal search experience that speaks the language of science — both verbally and mathematically.

### 7.2.2 Future Work

Although the current implementation is incredibly effective, numerous improvements can contribute to making the system more resilient and versatile in its applications:

**User Feedback Loop Integration:** By integrating a mechanism where user selection and action can be monitored and recorded, further search results improvement can be carried out in anticipation using reinforcement learning algorithms.

**Image-based Input Support:** The enhancement of the system to read written or scanned mathematics using image reading (OCR + equation parsing) can enhance the versatility of the system.

**Multilingual Support:** At present, the system is geared towards English-based scientific writing. Adding multilingual BERT variants would expand the system to reach a global community of researchers.

**Context-aware Re-ranking:** Adding document-level context (e.g., full-section embeddings or citation networks) to the re-ranking model can enable even more precise prioritization.

## References

*[1] A. Pathak, P. Pakray and R. Das, "LSTM Neural Network Based Math Information Retrieval," 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), Gangtok, India, 2019, pp. 1-6, doi: 10.1109/ICACCP.2019.8882887.*

*[2] Mali, A., Ororbia, A., Kifer, D., & Giles, C. L. (2021b, April 7). Recognizing and Verifying Mathematical Equations using Multiplicative Differential Neural Units. arXiv.org. https://arxiv.org/abs/2104.02899*

*[3] Patel, M. (2019, August 7). TinySearch -- Semantics based Search Engine using Bert Embeddings. arXiv.org. https://arxiv.org/abs/1908.02451*

*[4] Gao, G., Ju, H., Jiang, J., Qin, Z., & Dong, B. (2024, March 20). A semantic search engine for Mathlib4. arXiv.org. https://arxiv.org/abs/2403.13310*

*[5] Lukas Pfahler and Katharina Morik. 2020. Semantic Search in Millions of Equations. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20). Association for Computing Machinery, New York, NY, USA, 135–143. https://doi.org/10.1145/3394486.3403056*

*[6] Formal, T., Lassance, C., Piwowarski, B., & Clinchant, S. (2021). SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2109.10086*

*[7] Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., & Wu, J. (2019). The Neuro-Symbolic Concept Learner: Interpreting scenes, words, and sentences from natural supervision. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1904.12584*

*[8] Zhu, Z., Galkin, M., Zhang, Z., & Tang, J. (2022). Neural-Symbolic models for logical queries on knowledge graphs. arXiv (Cornell U University). https://doi.org/10.48550/arxiv.2205.10128*

*[9] Jun Xu, Xiangnan He, and Hang Li. 2019. Deep Learning for Matching in Search and Recommendation. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19). Association for Computing Machinery, New York, NY, USA, 832–833. https://doi.org/10.1145/3289600.3291380*

*[10] \F. Cakir, K. He, X. Xia, B. Kulis and S. Sclaroff, "Deep Metric Learning to Rank," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 1861-1870, doi: 10.1109/CVPR.2019.00196.*

*[11] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen and K. C. Tan, "A Survey on Evolutionary Neural Architecture Search," in IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 2, pp. 550-570, Feb. 2023, doi: 10.1109/TNNLS.2021.3100554. keywords: {Computer architecture;Optimization;Convolutional neural networks;Search problems;Neural networks;Deep learning;Statistics;Deep learning;evolutionary computation (EC);evolutionary neural architecture search (NAS);image classification},*

*[12] Ma, Y., Liu, J., Yu, M., Sun, Y., & Dong, X. (2021). LaPraDoR: Large-scale Pre Training for Dense Retrieval. arXiv preprint arXiv:2109.06271.*

*[13] Zhu, Y., Xu, J., & Wang, W. Y. (2020). TOME: Ontology-Based Semantic SearchPretrained Embeddings. Proceedings of ACL 2020.*

[14] *Peng, S., Ma, Y., Li, J., Wang, H., & Yin, Y. (2021). MathBERT: A Pretrained Model for Mathematical Formula Representation. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[15] *Ahmad, W. U., Zhong, H., Falk, R., & Chang, K. W. (2019). On Difficulties of Cross-Lingual Transfer with Order Differences: A Case Study on Dependency Parsing. arXiv preprint arXiv:1911.11108.*

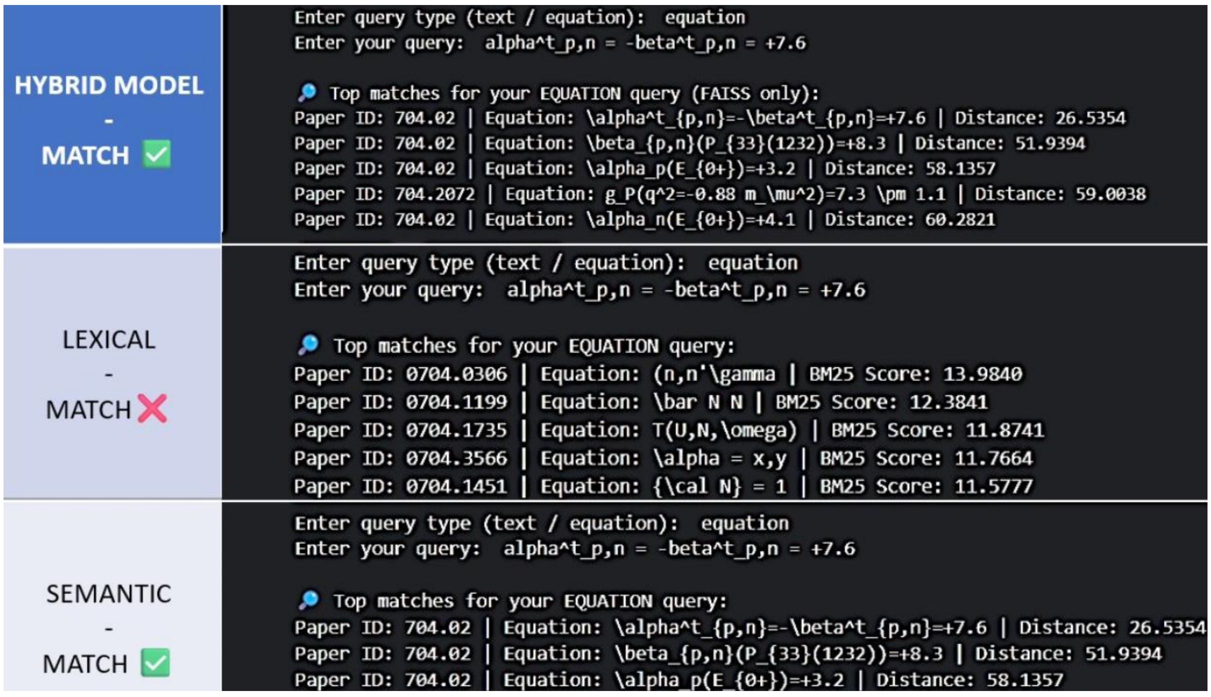
# Appendices

| | |
|---|---|
| FAISS | Facebook AI Similarity Search |
| NLTK | Natural Language Toolkit |
| BM25 | Best March 25 |
| SciBERT | Scientific Bidirectional Encoder Representations and Transformers |

## Screenshots

1. Experimental Screenshots of the model

   1.1 Equation Query

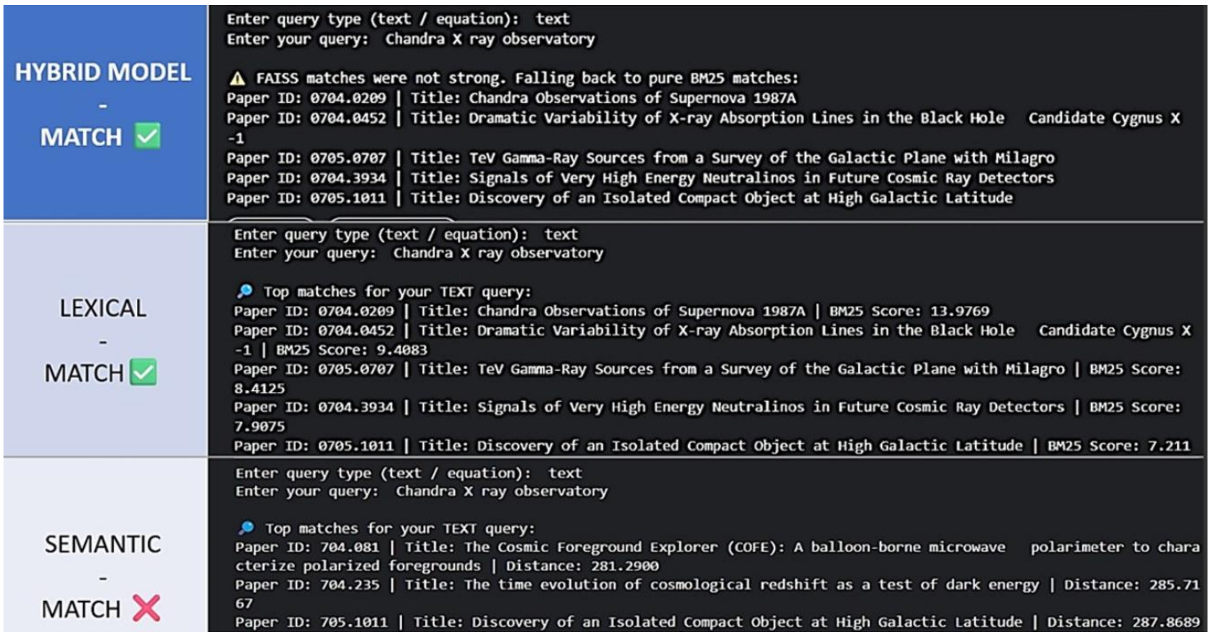


```
Enter query type (text / equation):  equation
Enter your query:  alpha^t_p,n = -beta^t_p,n = +7.6

🔎 Top matches for your EQUATION query (FAISS only):
Paper ID: 704.02 | Equation: \alpha^t_{p,n}=-\beta^t_{p,n}=+7.6 | Distance: 26.5354
Paper ID: 704.02 | Equation: \beta_{p,n}(P_{33}(1232))=+8.3 | Distance: 51.9394
Paper ID: 704.02 | Equation: \alpha_p(E_{0+})=+3.2 | Distance: 58.1357
Paper ID: 704.2072 | Equation: g_P(q^2=-0.88 m_\mu^2)=7.3 \pm 1.1 | Distance: 59.0038
Paper ID: 704.02 | Equation: \alpha_n(E_{0+})=+4.1 | Distance: 60.2821
```

```
Enter query type (text / equation):  equation
Enter your query:  alpha^t_p,n = -beta^t_p,n = +7.6

🔎 Top matches for your EQUATION query:
Paper ID: 0704.0306 | Equation: (n,n'\gamma | BM25 Score: 13.9840
Paper ID: 0704.1199 | Equation: \bar N N | BM25 Score: 12.3841
Paper ID: 0704.1735 | Equation: T(U,N,\omega) | BM25 Score: 11.8741
Paper ID: 0704.3566 | Equation: \alpha = x,y | BM25 Score: 11.7664
Paper ID: 0704.1451 | Equation: {\cal N} = 1 | BM25 Score: 11.5777
```

```
Enter query type (text / equation):  equation
Enter your query:  alpha^t_p,n = -beta^t_p,n = +7.6

🔎 Top matches for your EQUATION query:
Paper ID: 704.02 | Equation: \alpha^t_{p,n}=-\beta^t_{p,n}=+7.6 | Distance: 26.5354
Paper ID: 704.02 | Equation: \beta_{p,n}(P_{33}(1232))=+8.3 | Distance: 51.9394
Paper ID: 704.02 | Equation: \alpha_p(E_{0+})=+3.2 | Distance: 58.1357
```

## 1.2 Text Query



HYBRID MODEL - MATCH ✅

```
Enter query type (text / equation):  text
Enter your query:  Chandra X ray observatory

⚠ FAISS matches were not strong. Falling back to pure BM25 matches:
Paper ID: 0704.0209 | Title: Chandra Observations of Supernova 1987A
Paper ID: 0704.0452 | Title: Dramatic Variability of X-ray Absorption Lines in the Black Hole   Candidate Cygnus X
-1
Paper ID: 0705.0707 | Title: TeV Gamma-Ray Sources from a Survey of the Galactic Plane with Milagro
Paper ID: 0704.3934 | Title: Signals of Very High Energy Neutralinos in Future Cosmic Ray Detectors
Paper ID: 0705.1011 | Title: Discovery of an Isolated Compact Object at High Galactic Latitude
```

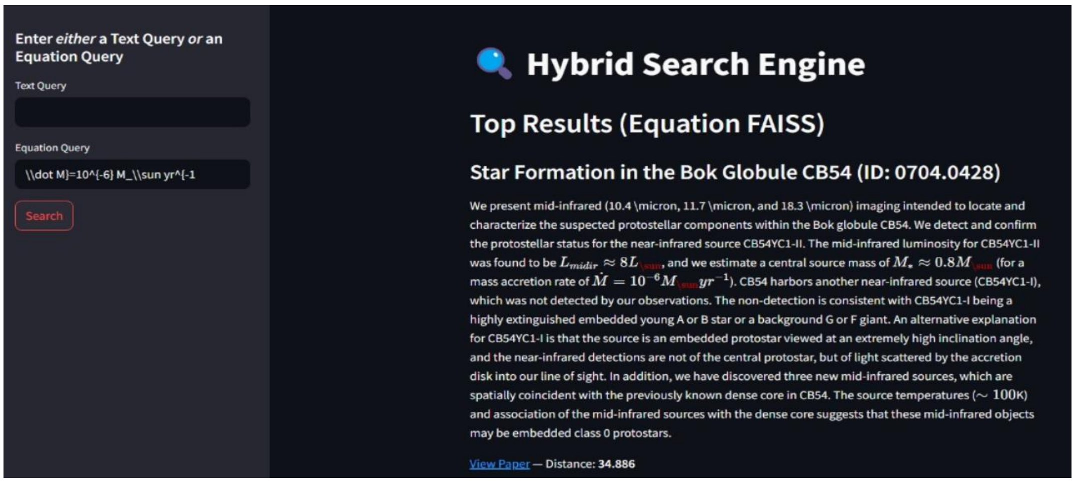
LEXICAL - MATCH ✅

```
Enter query type (text / equation):  text
Enter your query:  Chandra X ray observatory

🔎 Top matches for your TEXT query:
Paper ID: 0704.0209 | Title: Chandra Observations of Supernova 1987A | BM25 Score: 13.9769
Paper ID: 0704.0452 | Title: Dramatic Variability of X-ray Absorption Lines in the Black Hole   Candidate Cygnus X
-1 | BM25 Score: 9.4083
Paper ID: 0705.0707 | Title: TeV Gamma-Ray Sources from a Survey of the Galactic Plane with Milagro | BM25 Score:
8.4125
Paper ID: 0704.3934 | Title: Signals of Very High Energy Neutralinos in Future Cosmic Ray Detectors | BM25 Score:
7.9075
Paper ID: 0705.1011 | Title: Discovery of an Isolated Compact Object at High Galactic Latitude | BM25 Score: 7.211
```

SEMANTIC - MATCH ❌

```
Enter query type (text / equation):  text
Enter your query:  Chandra X ray observatory

🔎 Top matches for your TEXT query:
Paper ID: 704.081 | Title: The Cosmic Foreground Explorer (COFE): A balloon-borne microwave   polarimeter to chara
cterize polarized foregrounds | Distance: 281.2900
Paper ID: 704.235 | Title: The time evolution of cosmological redshift as a test of dark energy | Distance: 285.71
67
Paper ID: 705.1011 | Title: Discovery of an Isolated Compact Object at High Galactic Latitude | Distance: 287.8689
```

## 2. Model Results Including



Enter *either* a Text Query *or* an Equation Query

Text Query

Equation Query

\\dot M}=10^{-6} M_\\sun yr^{-1

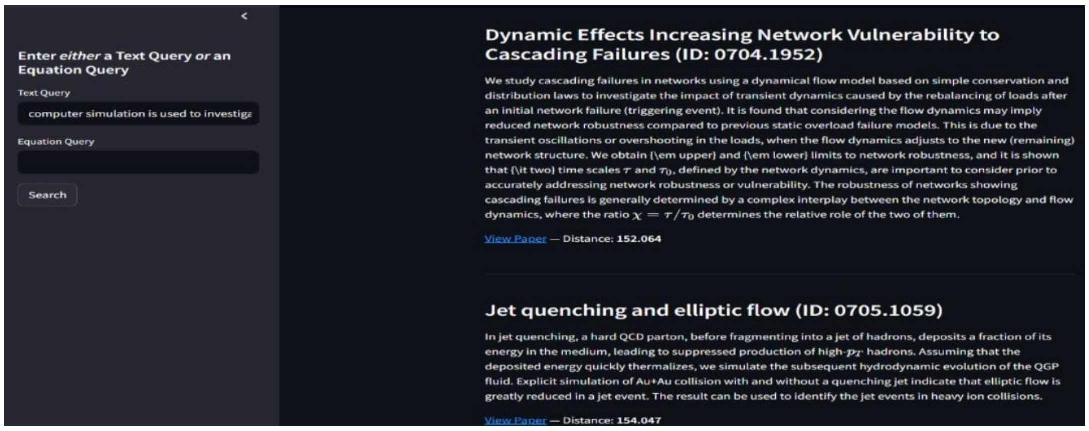
### 🔍 Hybrid Search Engine

### Top Results (Equation FAISS)

#### Star Formation in the Bok Globule CB54 (ID: 0704.0428)

We present mid-infrared (10.4 \micron, 11.7 \micron, and 18.3 \micron) imaging intended to locate and characterize the suspected protostellar components within the Bok globule CB54. We detect and confirm the protostellar status for the near-infrared source CB54YC1-II. The mid-infrared luminosity for CB54YC1-II was found to be $L_{midir} \approx 8L_{\sun}$, and we estimate a central source mass of $M_* \approx 0.8M_{\sun}$ (for a mass accretion rate of $\dot{M} = 10^{-6}M_{\sun}yr^{-1}$). CB54 harbors another near-infrared source (CB54YC1-I), which was not detected by our observations. The non-detection is consistent with CB54YC1-I being a highly extinguished embedded young A or B star or a background G or F giant. An alternative explanation for CB54YC1-I is that the source is an embedded protostar viewed at an extremely high inclination angle, and the near-infrared detections are not of the central protostar, but of light scattered by the accretion disk into our line of sight. In addition, we have discovered three new mid-infrared sources, which are spatially coincident with the previously known dense core in CB54. The source temperatures ($\sim 100$K) and association of the mid-infrared sources with the dense core suggests that these mid-infrared objects may be embedded class 0 protostars.

View Paper — Distance: **34.886**



Enter *either* a Text Query *or* an Equation Query

Text Query

computer simulation is used to investiga

Equation Query

#### Dynamic Effects Increasing Network Vulnerability to Cascading Failures (ID: 0704.1952)

We study cascading failures in networks using a dynamical flow model based on simple conservation and distribution laws to investigate the impact of transient dynamics caused by the rebalancing of loads after an initial network failure (triggering event). It is found that considering the flow dynamics may imply reduced network robustness compared to previous static overload failure models. This is due to the transient oscillations or overshooting in the loads, when the flow dynamics adjusts to the new (remaining) network structure. We obtain {\em upper} and {\em lower} limits to network robustness, and it is shown that {\it two} time scales $\tau$ and $\tau_0$, defined by the network dynamics, are important to consider prior to accurately addressing network robustness or vulnerability. The robustness of networks showing cascading failures is generally determined by a complex interplay between the network topology and flow dynamics, where the ratio $\chi = \tau/\tau_0$ determines the relative role of the two of them.

View Paper — Distance: **152.064**

#### Jet quenching and elliptic flow (ID: 0705.1059)

In jet quenching, a hard QCD parton, before fragmenting into a jet of hadrons, deposits a fraction of its energy in the medium, leading to suppressed production of high-$p_T$ hadrons. Assuming that the deposited energy quickly thermalizes, we simulate the subsequent hydrodynamic evolution of the QGP fluid. Explicit simulation of Au+Au collision with and without a quenching jet indicate that elliptic flow is greatly reduced in a jet event. The result can be used to identify the jet events in heavy ion collisions.

View Paper — Distance: **154.047**