A Major Project Report on

Modular Query Refinement Using Latent Dirichlet Allocation

Submitted in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Engineering in Software Engineering

under Pokhara University

Submitted by:

Binit Bikram KC, 201610

Dhiraj Poudel, 201612

Rishav Dahal, 201629

Sandhya Gotame, 201637

26 July 2025

Under the supervision of:

Mr. Ashim Khadka

Department of Software Engineering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur, Nepal

SUPERVISOR'S APPROVAL

This is to certify that the major project entitled "**Modular Query Refinement Using Latent Dirichlet Allocation**" undertaken and demonstrated by **Binit Bikram KC**, **Dhiraj Poudel**, **Rishav Dahal**, **Sandhya Gotame** has been successfully completed under my supervision as a partial fulfilment of the requirements for the degree of **Software Engineering** under Pokhara University. I, henceforth, approve this project to be awarded the certificate by the concerned authority.

During supervision, I found students hardworking, skilled and ready to undertake any professional work related to this field in future.

Ashim Khadka
Supervisor
Date: August 10, 2025

EXAMINERS' ACCEPTANCE

This is to certify that the major project entitled "**Modular Query Refinement Using Latent Dirichlet Allocation**" presented by **Binit Bikram KC**, **Dhiraj Poudel**, **Rishav Dahal**, **Sandhya Gotame** as a partial fulfilment of the requirements for the degree of **Software Engineering** under Pokhara University has been examined and accepted by the following panel of experts. We, henceforth, recommend this project to be awarded by the certificate from the concerned authority.

We extend all the best wishes to the students for their future careers.

Rupesh Kumar Nidhi
Examiner
Date: August 10, 2025

Hemraj Bhattarai
Examiner
Date: August 10, 2025

CERTIFICATE

Following the Supervisor's Approval and Examiners' Acceptance, the major project entitled "**Modular Query Refinement Using Latent Dirichlet Allocation**" submitted by **Binit Bikram KC**, **Dhiraj Poudel**, **Rishav Dahal**, **Sandhya Gotame** as a partial fulfilment of the requirements for the degree of **Software Engineering** under Pokhara University, has been officially awarded by this certificate.

I wish the students all the best for their future endeavours.

Er. Bhusan Thapa
Head, Department of Software Engineering
Date: August 10, 2025

DECLARATION

We, **Binit Bikram KC**, **Dhiraj Poudel**, **Rishav Dahal**, **Sandhya Gotame**, students of **Software Engineering**, Nepal College of Information Technology affiliated to Pokhara University, hereby declare that the work undertaken in this major project entitled "**Modular Query Refinement Using Latent Dirichlet Allocation**" is the outcome of our own effort and is correct to the best of our knowledge. This work has been accomplished by obeying the engineering ethics; and it contains neither materials published earlier or written by another person/people nor materials which has been accepted for the award of any other degree or diploma of the university or other institution, except where due acknowledgement has been made in the document.

(Binit Bikram KC)	**(Dhiraj Poudel)**
Student	Student
Date: August 10, 2025	Date: August 10, 2025

(Rishav Dahal)	**(Sandhya Gotame)**
Student	Student
Date: August 10, 2025	Date: August 10, 2025

Acknowledgment

We would like to express my sincere gratitude to all those who supported me throughout the development of this project.
First and foremost, we are deeply grateful to my supervisor, Dr. Ashim Khadka, for his continuous guidance, valuable insights, and encouragement throughout the project. His expert advice played a crucial role in shaping the direction of this work. We would also like to express our gratitude to our college for providing the environment necessary to make this project successful.
Our heartfelt thanks goes to our peers and friends for their constant support, constructive feedback, and for being a source of motivation during challenging phases of the project.

Abstract

Modular Query Refinement for Search Enhancement is a dynamic web-based application designed to improve the accuracy and efficiency of health-related searches. By leveraging natural language processing (NLP) techniques, the system extracts key terms from user queries and refines them to deliver more relevant search results.

The application processes each query through various refinement techniques, helping users get precise information faster by minimizing irrelevant results and enhancing the overall search experience. A core part of the system is its use of Latent Dirichlet Allocation (LDA)—an unsupervised learning model that identifies topics based on probabilistic distribution, allowing it to adapt flexibly to different types of hidden topics.

In addition to LDA, the system also explores other text encoding methods, including TF-iDF, Bag of Words, and BERT, comparing the result of those integrations to optimize keyword generation and improve the relevance of search responses. By refining queries based on contextual relevance, this approach helps search engines become more intelligent and user-focused.

***Keywords*— Web Application, Natural Language Processing, Latent Dirichlet Allocation (LDA), TF-iDF, BERT, Bag of Words**

Contents

List of Figures

List of Tables

1 Introduction

In today's information-driven world, users are often overwhelmed by vast volumes of digital content, making it increasingly difficult to retrieve relevant results from traditional search systems. Despite advancements in search technology, many engines still rely heavily on keyword matching, which fails to account for the nuances of human language and the complexity of user intent. This gap between user expectations and actual search results stems largely from vague or imprecise queries that do not fully capture the user's contextual needs or emotional undertone.

Modular Query Refinement for Search Enhancement addresses this challenge by introducing a smart, intermediary API that refines user queries before passing them to the target search engine or knowledge base. Using modular NLP components such as semantic analysis, Topic Modeling and clustering algorithms, the system enhances query clarity and relevance.

Its flexible design allows integration across a wide range of platforms and use cases, improving search efficiency and accuracy while reducing irrelevant results. By acting as a contextual filter between the user and backend systems, this API offers a scalable solution to improve the overall search experience.

2 Problem Statement

Traditional search engines often struggle in delivering relevant results because they rely heavily on basic keyword matching. This approach struggles to grasp the true intent behind a user's query, especially when the query is complex or specific to a particular domain. As a result, users are frequently presented with vague or irrelevant results that don't align with what they were actually looking for.

A key challenge is the inability of traditional systems to understand the semantics and context of queries. There's often a disconnect between how users phrase their questions and how information is indexed, leading to poor matches. These systems also lack the flexibility to adapt to different user needs or specialized domains, which contributes to longer search times and a frustrating user experience.

3 Project Objectives

This project aims to develop an intelligent and modular system that enhances the effectiveness of search engines by improving how user queries are interpreted and processed. Through the integration of Natural Language Processing techniques, the system seeks to generate more accurate and meaningful queries tailored to both context and intent.

- To design and implement a modular query expansion system using Natural Language Processing (NLP) techniques.
- To accurately interpret the user's intent behind search queries for improved understanding.
- To refine and expand queries to enhance the relevance of search results.
- To adapt query processing dynamically based on domain-specific context and user preferences.

4 Significance of the Study

The ”Modular Query Refinement for Search Enhancement” project holds significant importance in the domains of information retrieval, natural language processing, and user experience optimization, offering several practical benefits and applications:

- Helps bridge the gap between user intent and actual search engine results through semantic understanding.
- Enhances the quality and accuracy of retrieved information, especially in complex or domain-specific queries.
- Reduces user frustration by minimizing irrelevant or ambiguous results from poorly phrased queries.
- Introduces a modular architecture that allows flexibility and scalability for future improvements.
- Promotes efficient and intelligent search interactions, saving time and improving productivity.
- Encourages adoption of NLP in real-world search systems, demonstrating the value of AI in everyday information retrieval tasks.

5 Scope and Limitations

5.1 Scope:

This project focuses on enhancing search systems by refining user queries through advanced Natural Language Processing (NLP) techniques. The system aims to improve the accuracy and relevance of search results across both general and domain-specific applications by restructuring queries based on user context and intent.

1. Refines user queries using NLP to produce more meaningful and context-aware search inputs.
2. Enhances search performance for both general use cases and specialized domains like healthcare, education, or legal research.
3. Uses modular components (e.g., preprocessing,keyword expansion) that can be independently extended or replaced.
4. Designed for easy integration with existing search engines, knowledge bases, or recommendation systems.
5. Can act as a middleware layer between users and different search platforms, improving relevance without modifying backend systems.
6. Web-based architecture ensures accessibility across devices and environments.
7. Offers potential for future expansion to support multiple languages and domain-specific adaptations.

5.2 Limitations:

While the system significantly improves query refinement, there are certain constraints related to model performance, domain adaptability, and system integration. These limitations help define the current operational boundaries of the project. following limitations:

1. NLP model performance depends on the quality and coverage of the training data.
2. Queries that are too vague or too short may not benefit significantly from refinement.
3. Domain-specific accuracy requires fine-tuning with relevant datasets, which may not always be readily available.
4. Integration with some closed or proprietary search engines may be technically or legally restricted.
5. Real-time performance may degrade under high-volume or resource-constrained environments without further optimization.
6. The system focuses on refining the input query only and does not alter the internal ranking mechanisms of the search engine.

6 Literature Review

Query expansion (QE) is a core technique in information retrieval (IR) aimed at improving search accuracy by enhancing the original user query with additional relevant terms. It has evolved significantly since the early days of computer-based search systems.

The concept of query expansion to enhance search effectiveness was first introduced in 1960 by Melvin E. Maron and John L. Kuhns in the paper On Relevance, Probabilistic Indexing and Information Retrieval [1].This paper reports on a novel technique for literature indexing and searching in a mechanized library system. It proposed the weighted index tags replacing traditional binary index-term assignment. It introduced the concept of a ”relevance number”, computed for each document relative to a query, allowing documents to be ranked probabilistically rather than retrieved via strict Boolean matching [1].

Different information retrieval techniques were in rapid development, aimed at determining and retrieving information. The groundwork for effective search started during the 1970s, in which primitive methods like basic keyword matching and manual indexing of information were performed. The concept of Vector Space Model emerged during this time introducing ranked retrieval based on geometric similarity [2]. And later in 1972 the concept of Tf-iDf was introduced to avoid the shortcoming of the vector space model in which term frequency and inverse document frequency is measured and the relevancy of the word is determined [3].

$$\textbf{TF-IDF}(\text{t,d}) = \text{TF}(\text{t,d}) \times \log \frac{N}{DF(t)}$$

Latent Dirichlet Allocation (LDA) was introduced by Blei, Ng and Jordan in 2003 as a probabilistic model for topic modeling [4]. Latent Dirichlet Allocation (LDA) is a generative model used to discover hidden topics within a collection of documents. It assumes that each document is made up of a mixture of topics, and each topic consists of a set of words that are likely to appear together. Instead of assigning a single topic to a document, LDA allows documents to contain multiple topics in different proportions. LDA uses two Dirichlet distributions: one at the document level to represent the distribution of topics in a document, and another at the topic level to represent the distribution of words within each topic. By analyzing patterns of word usage across the text, LDA estimates the probability distributions that best explain how the words and topics are organized. This makes it a powerful and flexible method for understanding the underlying themes and structure in large text datasets.

Latent Semantic Indexing (LSI), also known as Latent Semantic Analysis (LSA), is another influential technique in the field of information retrieval that emerged in the late 1980s. LSI aims to improve search accuracy by capturing the hidden relationships between terms and documents through dimensionality reduction. It uses a mathematical technique called Singular Value Decomposition (SVD) to reduce the large term-document matrix into a lower-dimensional space, where semantically similar terms and documents are mapped closer together [5]. It decomposes the matrix into three new matrices: the singular value matrix, the words matrix, and the documents matrix [6]. By using LSA, we can calculate the similarity between words or documents based on the cosine distance between their representation vectors in latent space. LSA allows us to find words that have similar meanings or documents that have similar topics based on the similarity of semantic patterns found through SVD decomposition [5].

7 Methodology

7.1 Software Development Life Cycle (SDLC)

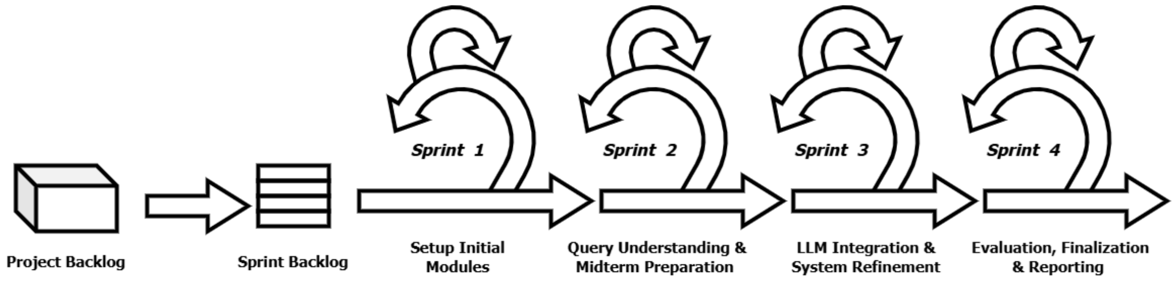

Sprint 1
Sprint 2
Sprint 3
Sprint 4
Project Backlog
Sprint Backlog
Setup Initial Modules
Query Understanding & Midterm Preparation
LLM Integration & System Refinement
Evaluation, Finalization & Reporting

Figure 1: Agile Process

This project follows the Agile Software Development Life Cycle (SDLC) with a focus on the Scrum framework, ensuring iterative development, regular feedback, and adaptability to changing requirements. The methodology supports the dynamic nature of Natural Language Processing (NLP)-based systems, such as our Modular Query Refinement Tool.

1. **Sprint 1: Setup and Initial Modules (Weeks 1–2)**
   The first sprint focused on setting up the development infrastructure and building a functional pipeline for user interaction. The frontend was developed to handle basic user input through a search bar, which connects to a REST API implemented in the backend. Meanwhile, work began on the NLP side with the identification and selection of a medical QnA dataset. Several preprocessing experiments were conducted to determine the best method for keyword extraction, particularly by extracting keywords from the answer part of the dataset.
   Key tasks:

   - Build a basic frontend UI that accepts user queries.
   - Implement a functional REST API for backend communication.
   - Select and finalize a medical QnA dataset.
   - Test and compare three approaches for extracting keywords from answers.
   - Begin structuring dataset into input-question → output-keywords format.

2. **Sprint 2:Dataset Finalization and Midterm Preparation (Weeks 3-4)**
   This phase centers around completing the preprocessing pipeline and preparing training data for the LDA model. Based on the experiments conducted in the first sprint, a final keyword extraction strategy will be selected. The focus is to complete a working dataset and demonstrate a functional pipeline during the midterm review. Additionally, the frontend and backend integration will be refined, and architectural documentation will be prepared for the midterm presentation.
   Key tasks:

   - Finalize the keyword extraction approach and generate full training dataset.
   - Polish REST API interactions and ensure proper data flow.
   - Prepare and document the overall system architecture.
   - Demonstrate data flow from query to structured format with sample outputs.
   - Prepare midterm deliverables: code walkthrough, diagrams, and demo-ready UI.

3. **Sprint 3:Model Training and Integration (Weeks 5–6)**
   This sprint focuses on training different models using the structured dataset created in Sprint 2. Once trained, the model will be wrapped and exposed via the backend for real-time inference. Backend logic will be updated to include keyword generation, and frontend integration will be enhanced to display model outputs. This sprint also involves testing model accuracy using internal validation techniques and ensuring modularity in model deployment.
   Key tasks:

   - Train different models on the prepared keyword dataset.
   - Create a model inference pipeline and integrate it with the backend.
   - Refactor backend code to include model response handling.
   - Update the frontend to display model-generated keywords or suggestions.
   - Perform internal evaluations on model performance.

4. **Sprint 4:UI Enhancement, Testing, and Finalization (Weeks 7–8)**

   The final sprint is dedicated to polishing the frontend UI and ensuring complete system integration. User experience will be improved with a cleaner design and better interactivity. Rigorous testing will be performed on all components to prepare for the final demo. Additionally, documentation and the project report will be compiled, covering methodology, experiments, results, and system design.

   Key tasks:

   - Enhance the frontend UI for better visual appeal and usability.
   - Conduct system-wide integration tests and resolve bugs.
   - Finalize the backend-model integration and response formats.
   - Write the final project report and prepare presentation material.
   - Ensure the system is demo-ready with backup workflows if needed.

7.2 Tools and Technique

- **Frontend:**

  The frontend of the system was developed using Nuxt.js, a powerful Vue.js framework for building modern web applications. It allows seamless integration with RESTful APIs for submitting user queries and displaying refined results in real time. Tailwind CSS was used for styling the interface, enabling a clean and responsive design system with utility-first classes. The combination of Nuxt.js and Tailwind ensured both maintainability and performance across different devices.

- **Backend(Django):**

  The backend of the system was developed using Django, a high-level Python web framework. Django REST Framework (DRF) was used to build the API endpoints that connected the frontend to the keyword extraction and query refinement logic. Its modularity allowed for easy extension and integration with the machine learning components.

- **Natural Language Processing (NLP) Libraries):**

  For core text preprocessing, libraries like NLTK and spaCy were used. These tools handled essential tasks such as tokenization, lemmatization, removal of stopwords, and part-of-speech tagging. Such preprocessing steps were necessary to prepare the raw medical dataset for effective keyword extraction and topic modeling.

- **Latent Dirichlet Allocation (LDA)**:
  LDA was used as one of the primary topic modeling techniques in the project. It identifies latent topics within a corpus by estimating word-topic and topic-document distributions. Implemented using the Gensim library, LDA helped group related medical terms and phrases, making it easier to interpret user queries in a domain-specific context. The model's performance was evaluated using coherence scores to determine the most meaningful topic structure.

- **Latent Semantic Indexing (LSI)**:
  LSI was also used for topic modeling but relied on Singular Value Decomposition applied to a TF-IDF matrix. Unlike LDA, which is probabilistic, LSI is based on linear algebra and helps capture latent semantic relationships between terms and documents. Although slightly less interpretable than LDA, LSI provided valuable insights when combined with statistical term weighting.

8 System Model and UML Diagram

8.1 Usecase Diagram

A use case diagram shown in Figure **??** depicts the interactions between users (actors) and a system to achieve specific goals. It shows the relationships between actors and use cases, highlighting the functionalities provided by the system from the user's perspective.

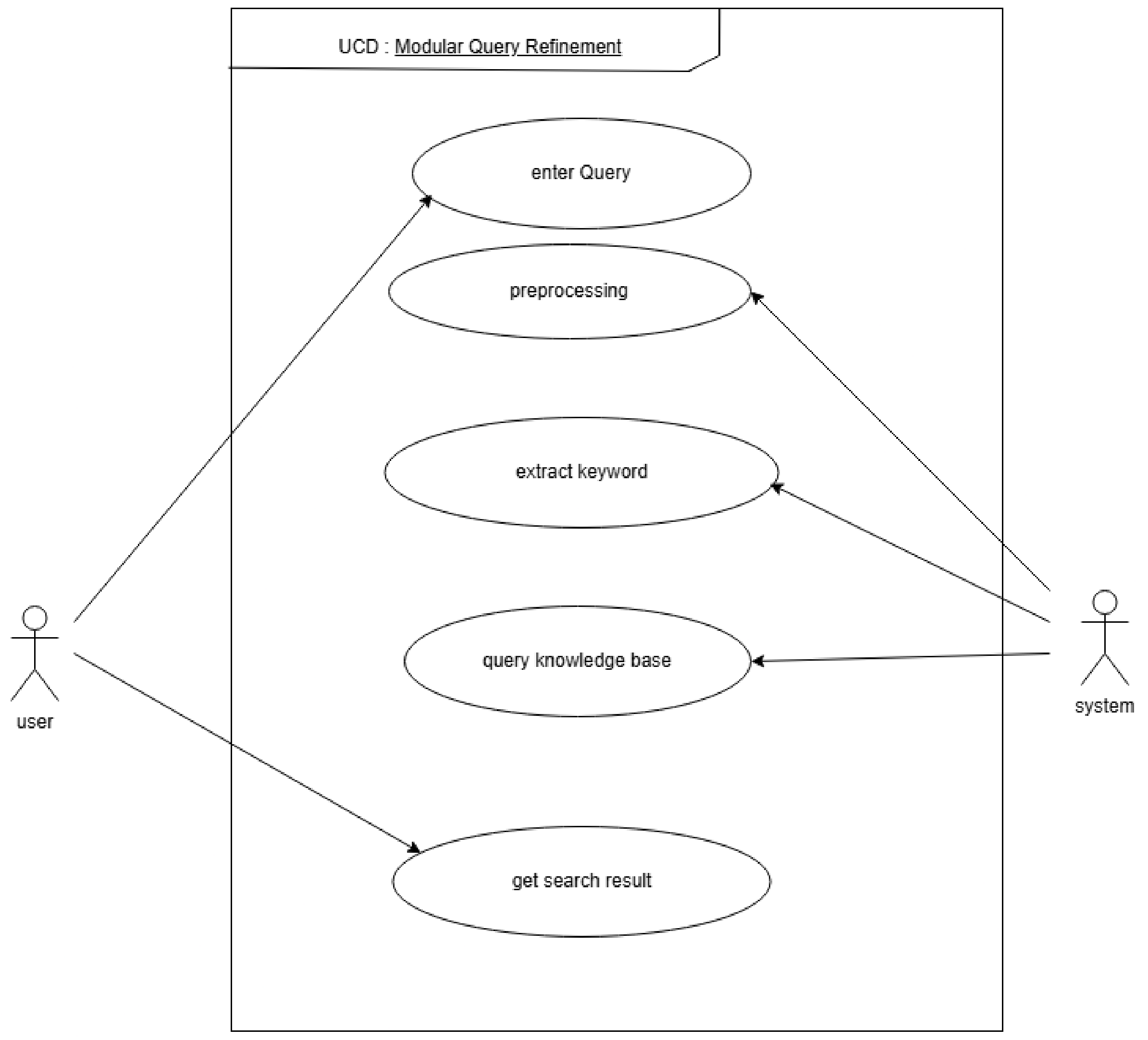


Figure 2: Usecase Diagram of Modular Query Refinement where there are two actors

8.2 Activity Diagram

An activity diagram illustrates the flow of control within a system or a specific process. It shows the sequence of activities, actions, and decisions involved in completing a task or achieving a goal. It's useful for modeling business processes or the logic of algorithms.

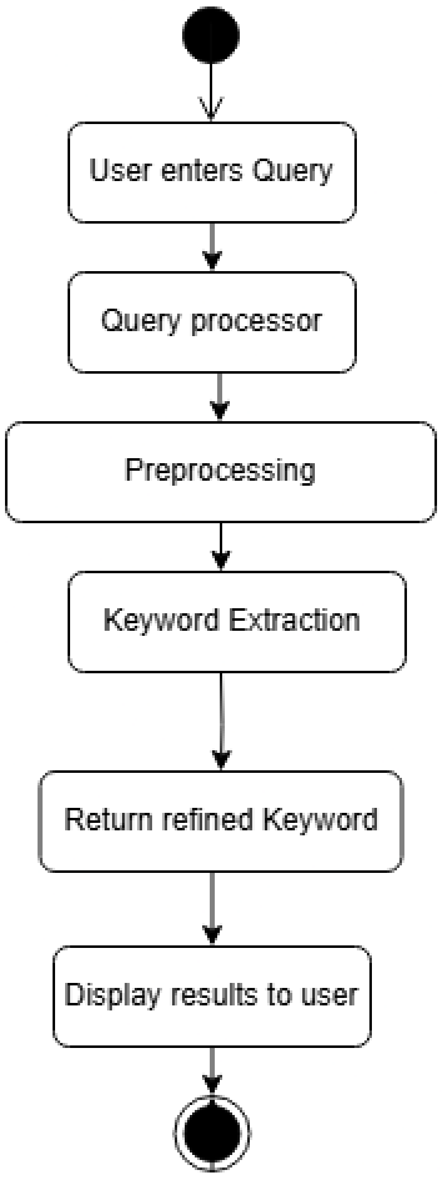


Figure 3: Activity Diagram

8.3 Component Diagram

A component diagram shows the organization and dependencies among software components in a system. It depicts the physical components (like executable files, libraries, etc.) and their relationships, providing a high-level view of the architecture.

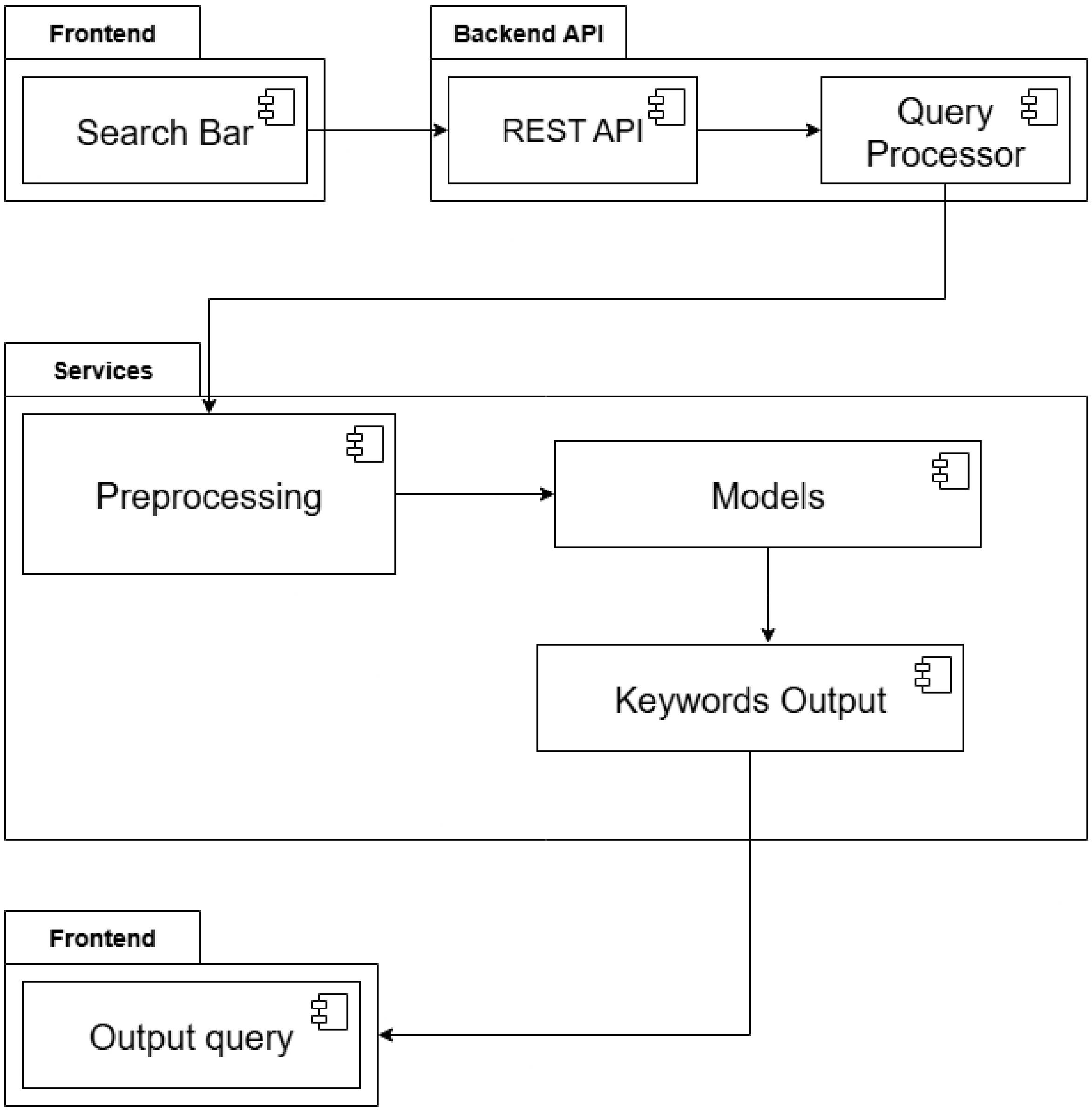


Figure 4: Component Diagram

8.4 System sequence Diagram

A system sequence diagram (SSD) represents the sequence of interactions between external actors (users or other systems) and a system to accomplish a specific functionality. It emphasizes the message flows and their order between objects over time.

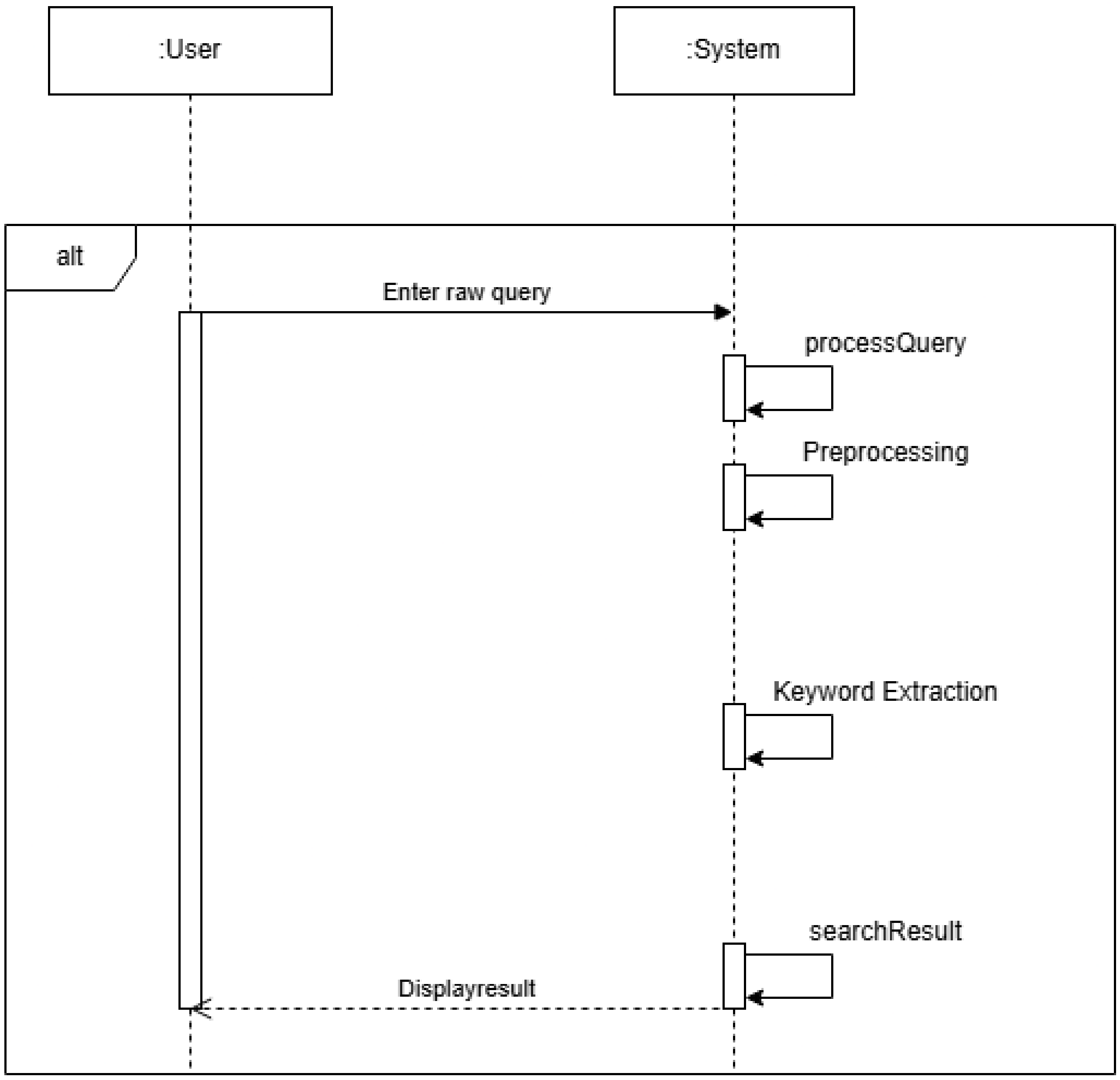


Figure 5: System sequence Diagram

9 Performance Evaluation

In our study, we explored four different methods to extract topics from the MedQuAD QnA dataset. Among these, we primarily focused on two approaches that yielded the most interpretable and coherent results:

1. Training an LDA Model with Coherence Values.
2. Training an LSI Model with TF-IDF.

9.1 LDA Model with Coherence Values

This approach aimed to find the optimal number of topics by measuring **coherence scores**—an evaluation metric that indicates how semantically interpretable the generated topics are.

9.1.1 Dataset Preparation

We used the **MedQuAD QnA dataset** and created a new column by concatenating each question with its corresponding answer. From this combined column, we generated a dictionary and corpus using the *Bag-of-Words (BoW)* method. This corpus was then used to train multiple **LDA (Latent Dirichlet Allocation)** models by varying the number of topics from 5 to 50.

9.1.2 Coherence Score Analysis

For each model, we computed the **coherence score** to determine the quality of the topics. The relationship between the number of topics and the coherence values is shown in the graph below:

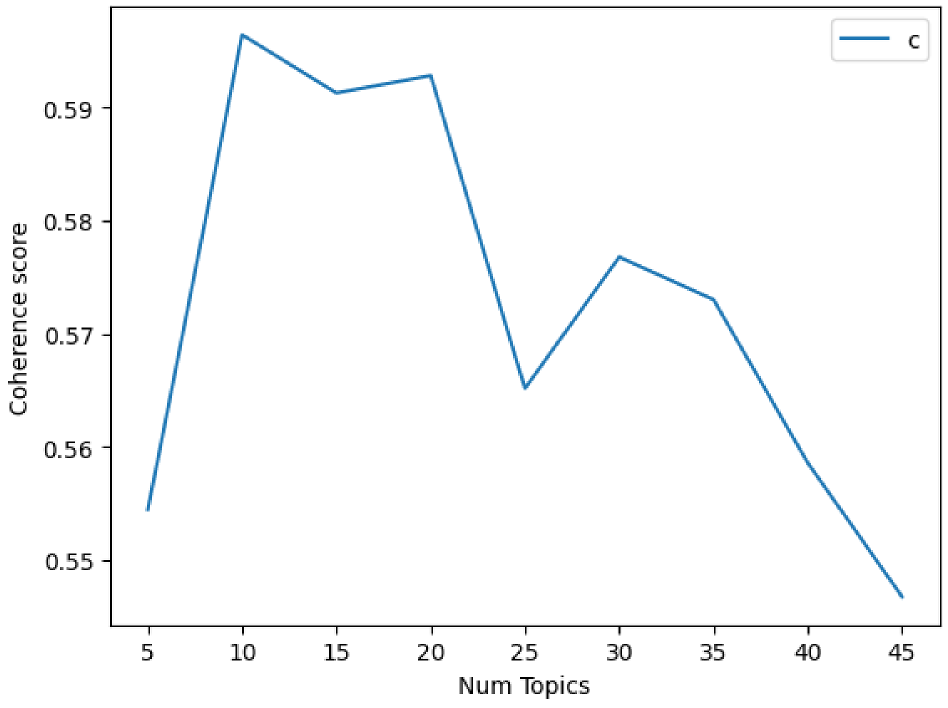


Figure 6: Coherence Score vs. Number of Topics

Figure 6 shows that the optimal number of topics selection as **10**, where the coherence value peaked. We then used the model trained with 10 topics for further analysis.

9.1.3 Topic Interpretation

The generated topics from the best LDA model are shown below:

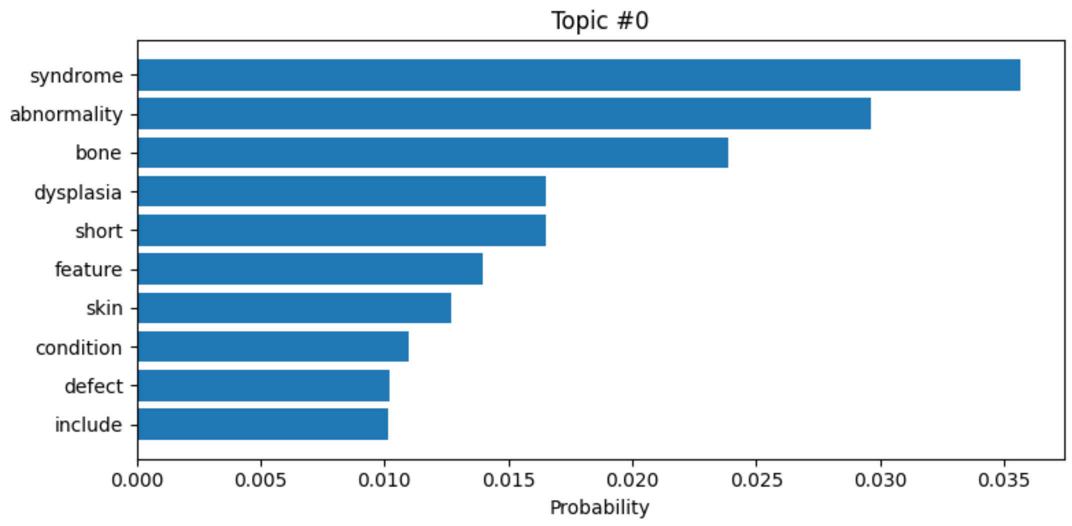


Figure 7: Topic00-LDA

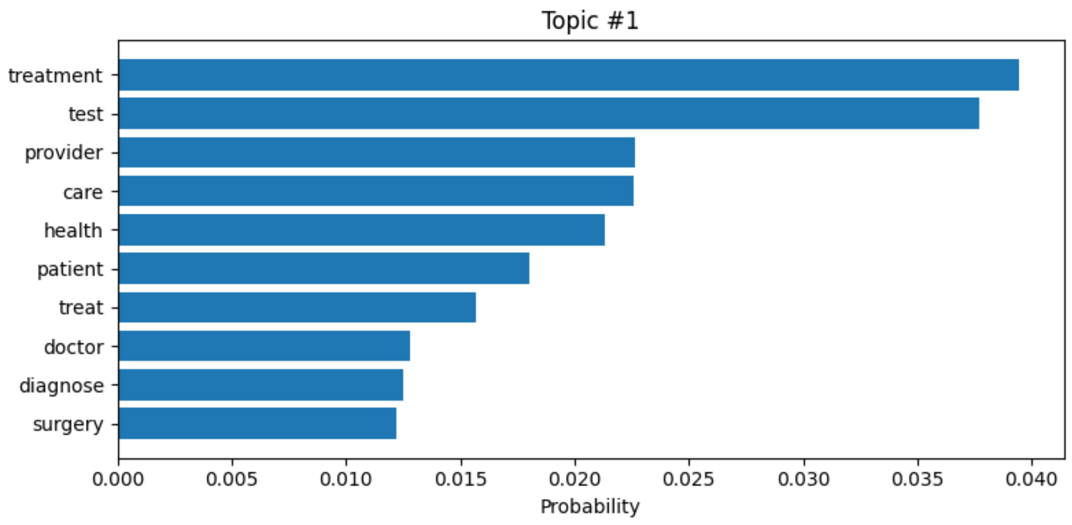


Figure 8: Topic01-LDA

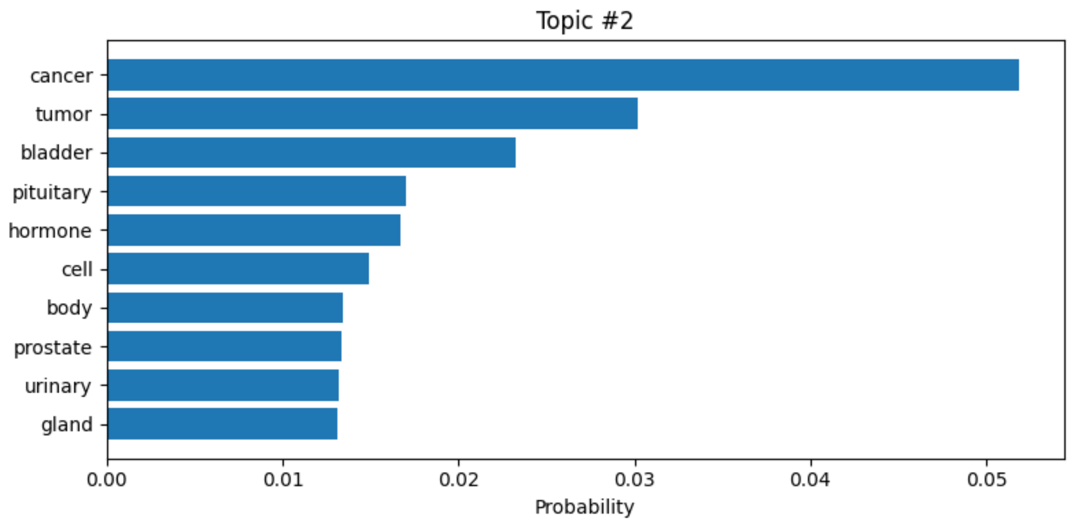


Figure 9: Topic02-LDA

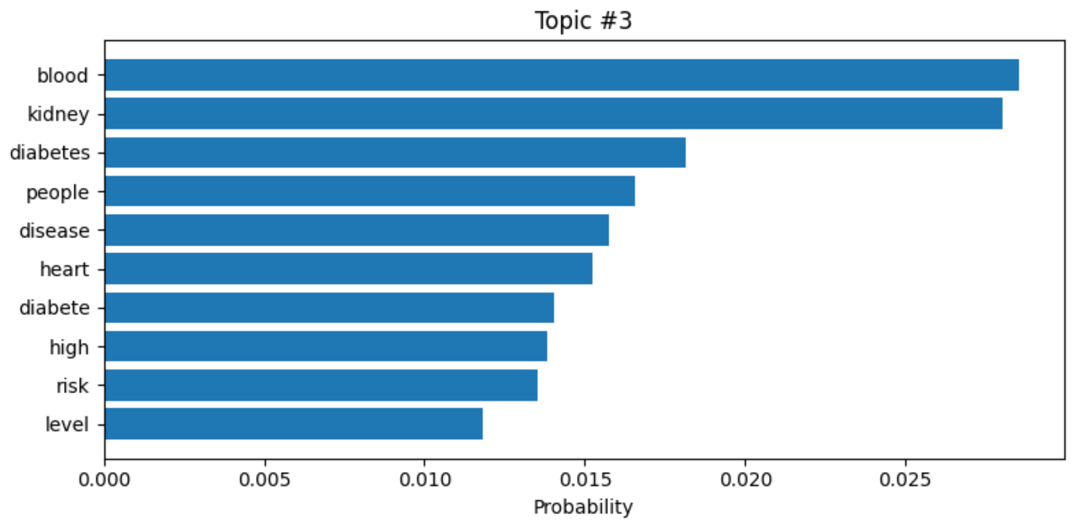


Figure 10: Topic03-LDA

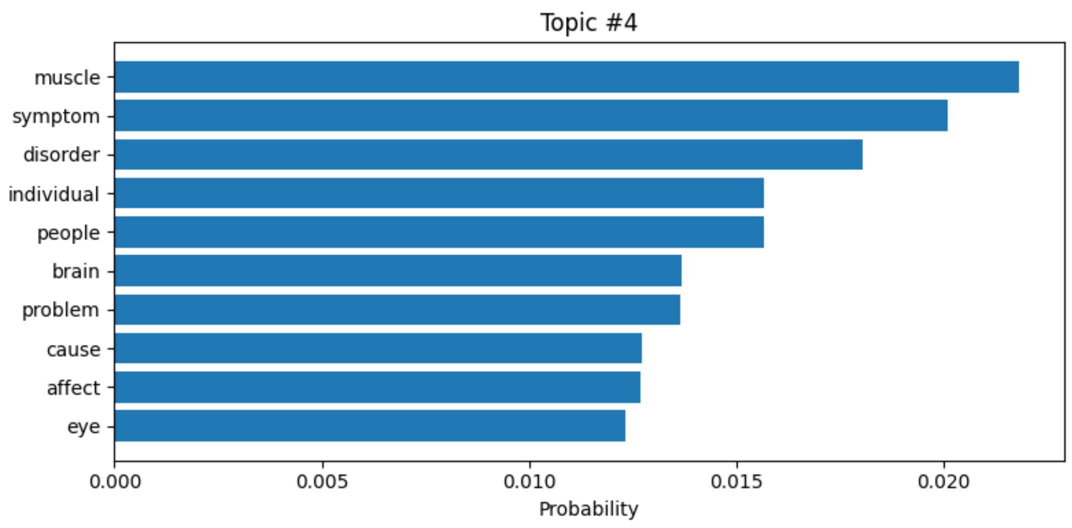


Figure 11: Topic04-LDA

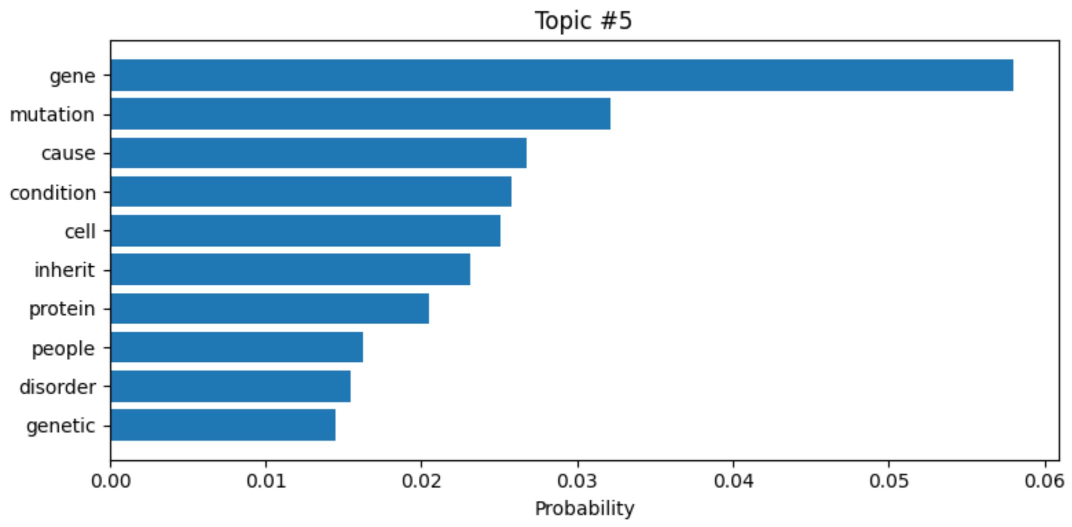


Figure 12: Topic05-LDA

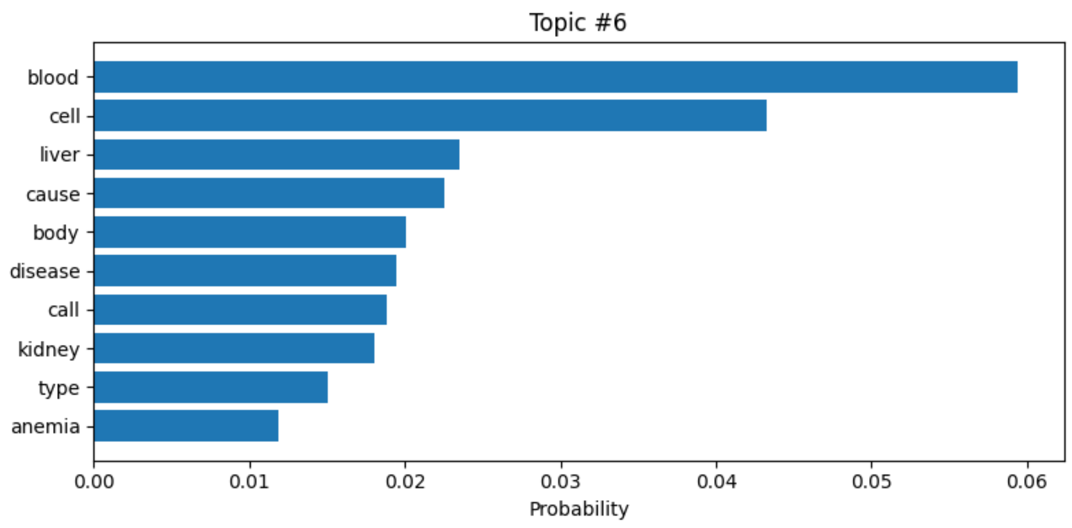


Figure 13: Topic06-LDA

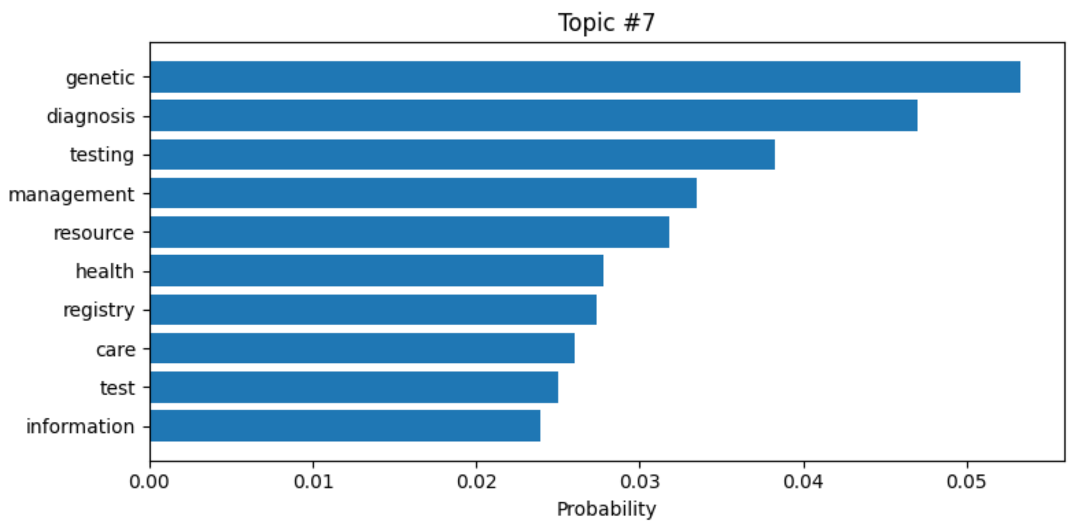


Figure 14: Topic07-LDA

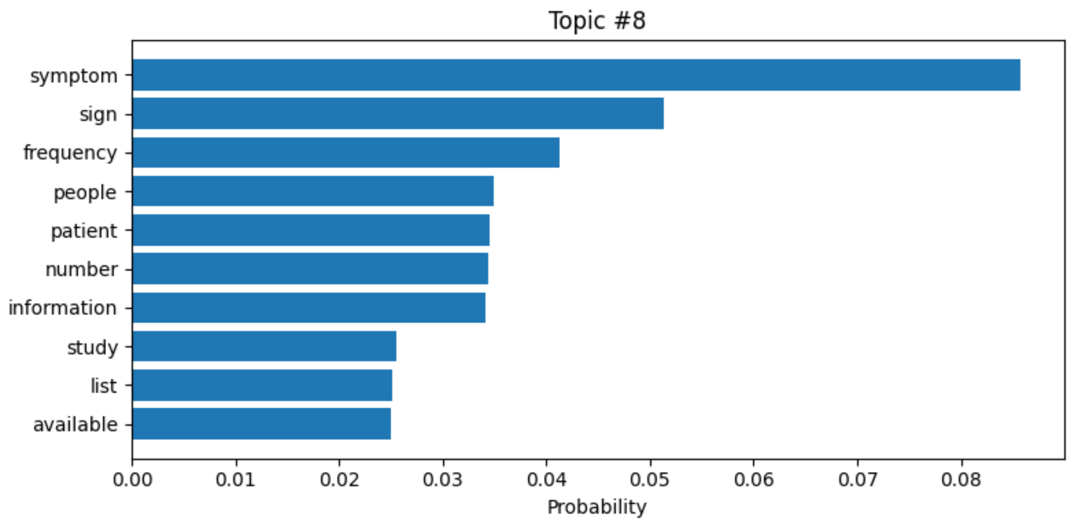


Figure 15: Topic08-LDA

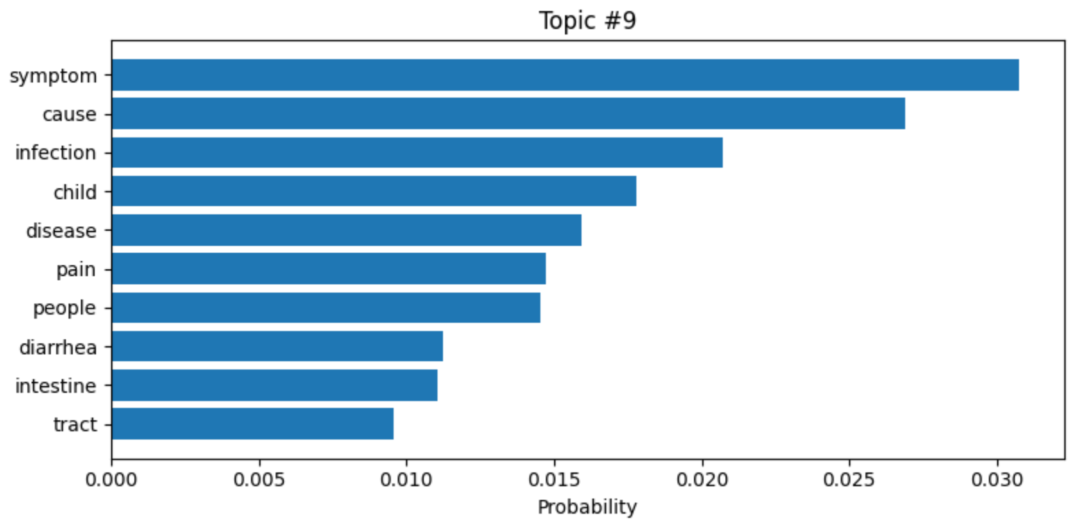


Figure 16: Topic09-LDA

These topics covered a wide range of medical themes and showed high interpretability upon manual inspection.

9.1.4 Example Query

To illustrate how the model responds to user queries, we provide the following example:

- **Query:** *”pain in urinary tract”*
- **Generated Tokens:** [symptom, cause, infection, child, disease, pain, people, diarrhea, intestine, track]

The model then outputs the topic distribution for the query as shown below:

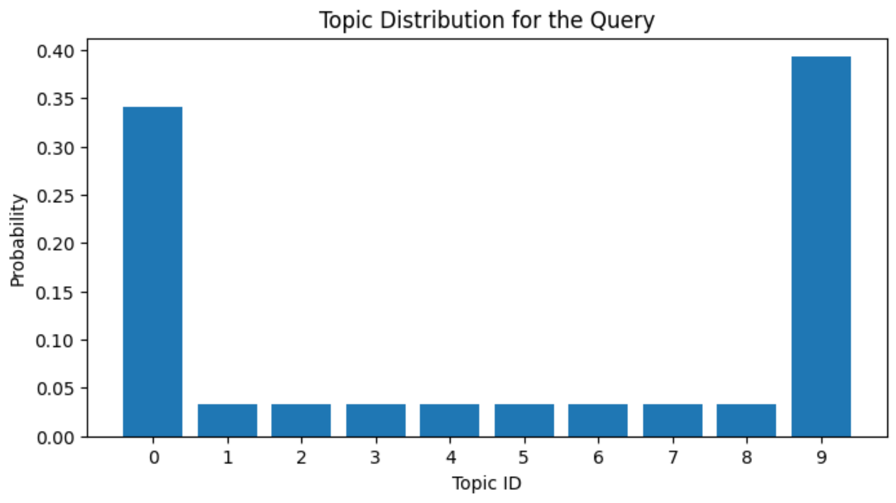


Figure 17: Topic Distribution for Sample Query

9.2 LSI Model with TF-IDF

In contrast to LDA, the **Latent Semantic Indexing (LSI)** model uses a **TF-IDF** representation instead of Bag-of-Words (BoW). The same preprocessed corpus was used, but the focus here was on dimensionality reduction and identifying latent topics through *Singular Value Decomposition (SVD)*.
We trained LSI models for different numbers of topics and evaluated them using the same coherence metric. The highest coherence score was achieved with **5 topics**, indicating that the optimal representation in the LSI space had fewer dimensions compared to the LDA model.

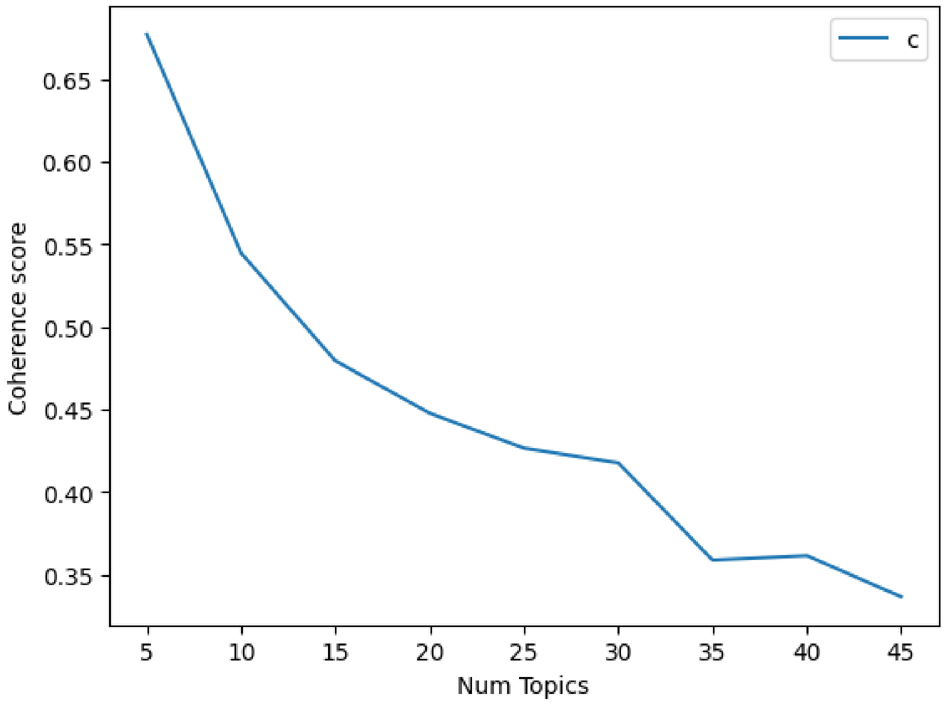


Figure 18: Coherence Score vs. Number of Topics for LSI

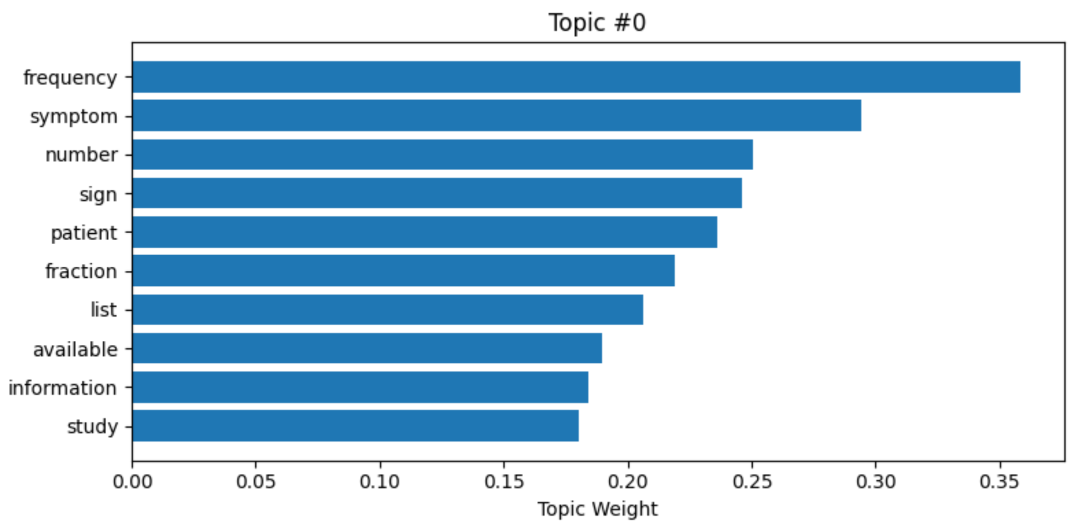


Figure 19: Topic00-LSI

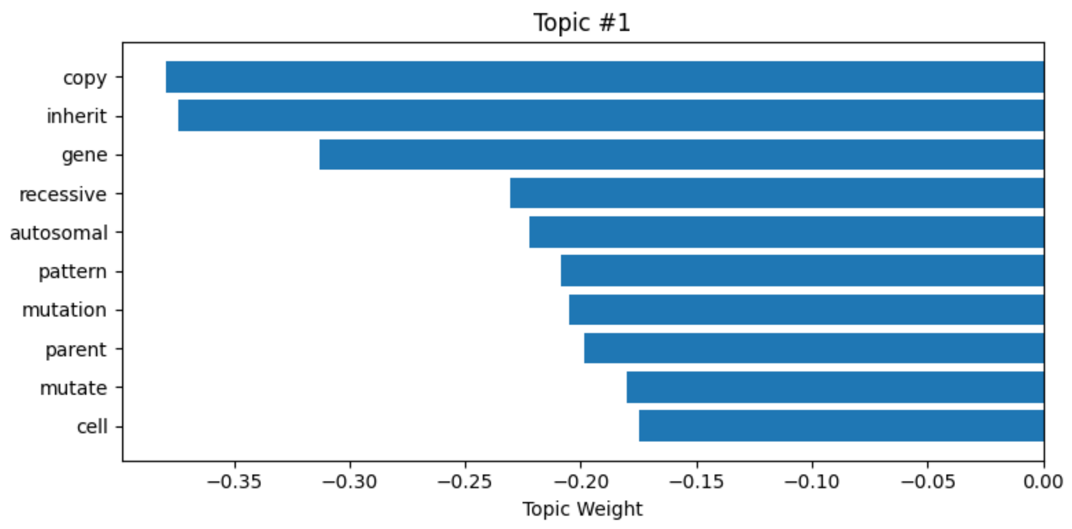


Figure 20: Topic01-LSI

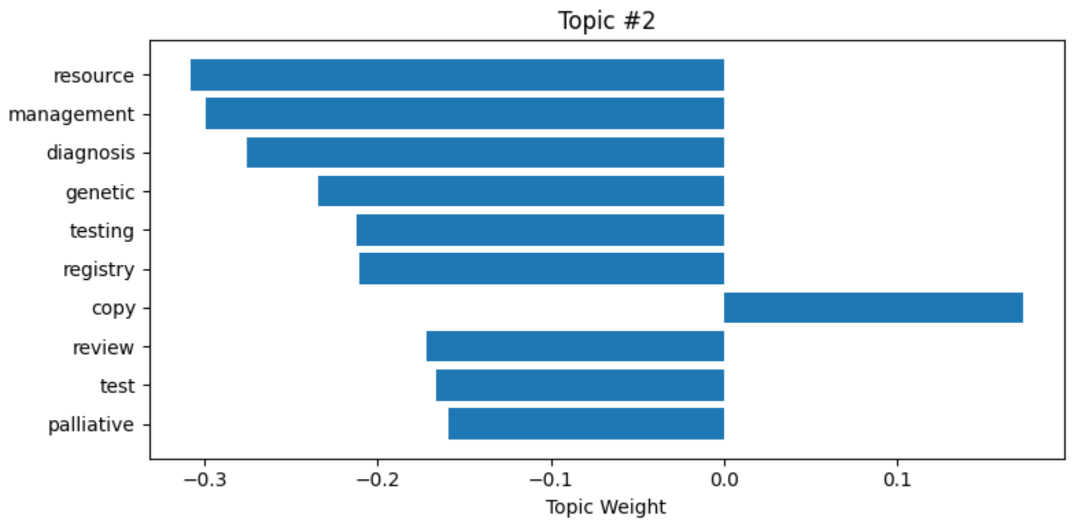


Figure 21: Topic02-LSI

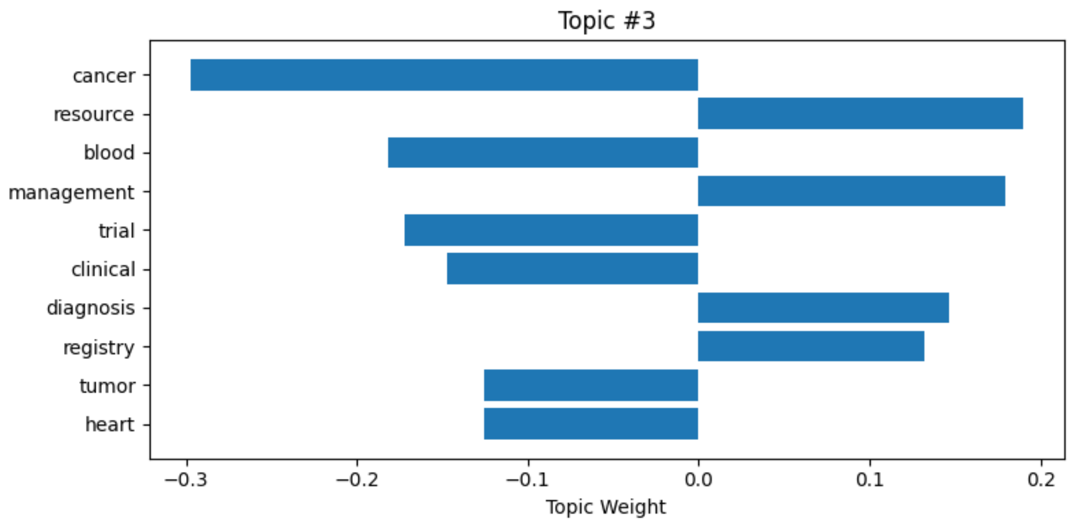


Figure 22: Topic03-LSI

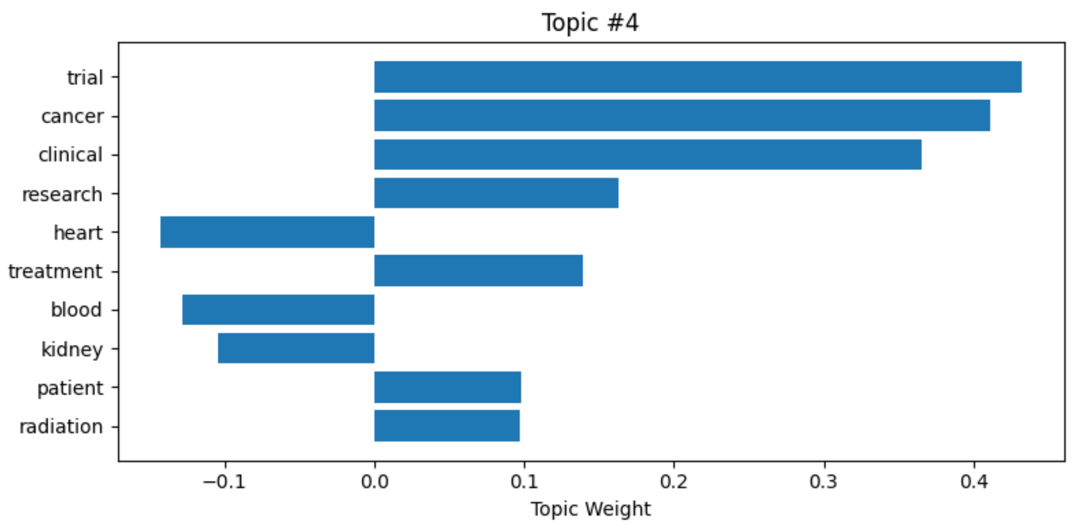


Figure 23: Topic04-LSI

While LSI performed reasonably well, the topics were slightly more abstract and less interpretable compared to LDA. This suggests that LDA may be more suitable for tasks requiring fine-grained semantic separation.

9.2.1 Example Query

To illustrate how the model responds to user queries, we provide the following example:

- **Query:** *"pain in urinary tract"*
- **Generated Tokens:** [cancer, resource, blood, management, trail, clinical, diagno-sis, registtry, tumor, heart]

The model then outputs the topic distribution for the query as shown below:

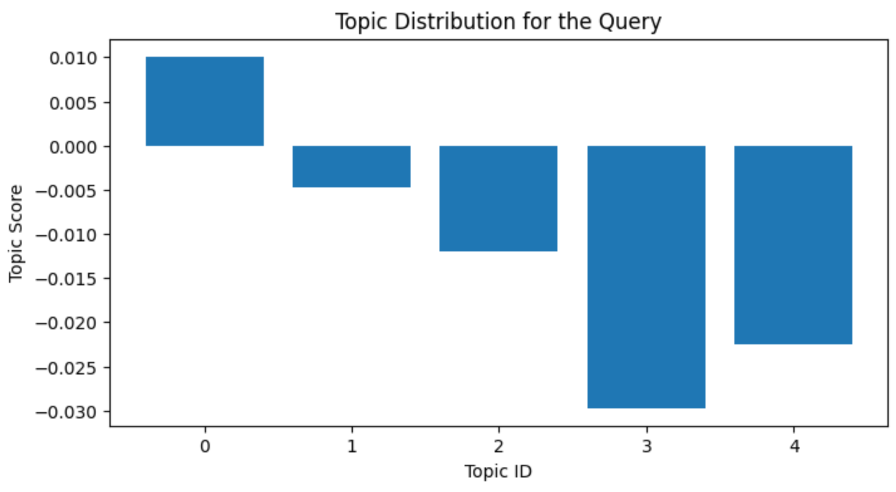


Figure 24: Topic Distribution for Sample Query

10 Results and Discussion

All four Models

Query = ”pain in urinary track”

The four algorithms LDA, LDA with verb filtering, LSI with TF-IDF, and BERT demonstrated varying capabilities in interpreting the query ”I have pain in urinary tract.” LDA produced a broad set of medically related keywords, but often included loosely connected terms, indicating limited contextual understanding. LDA with verb filtering slightly improved the relevance by focusing more on nouns, reducing noise and capturing more domain-specific keywords like ”infection” and ”bladder.” LSI with TF-IDF emphasized statistically significant terms such as ”diagnosis,” ”clinical,” and ”tumor,” showing its strength in identifying structured, document-level associations. BERT, however, provided the most semantically coherent and context-aware results, accurately linking ”urinary,” ”stone,” ”dialysis,” and ”disease” in a way that closely aligns with the intent and medical context of the original query. This comparison highlights how deep learning models like BERT excel in capturing nuanced meanings, especially in domain-specific or symptom-based queries.

LDA	**LDA with Verb**	**LSI with TF-IDF**	**BERT**
kidney, urine, bladder, urinary, eye, tract, urethra, stone, blood, vision	symptom, cause, infection, child, disease, pain, people, diarrhea, intestine, tract	cancer, resource, blood, management, trial, clinical, diagnosis, registry, tumor, heart	kidney, urine, bladder, blood, urinary, stone, cause, disease, tract, dialysis

Table 1: Topic keywords extracted by different models

11 Project Task and Time schedule

The distribution of project tasks, their scheduling , and the assignment of responsibilities among team members can be depicted through the utilization of the subswquent table and Gantt chart

11.1 Team Members And Their Responsibilities

Name	**Responsibility**
Rishav Dahal	Backend Logic and Semantic analysis
Dhiraj Poudel	Topic Modeling
Binit Bikram Kc	Frontend and API Integration
Sandhya Gautam	Bert model training

Table 2: Team Members and Their Responsibilities

11.2 Gantt Chart

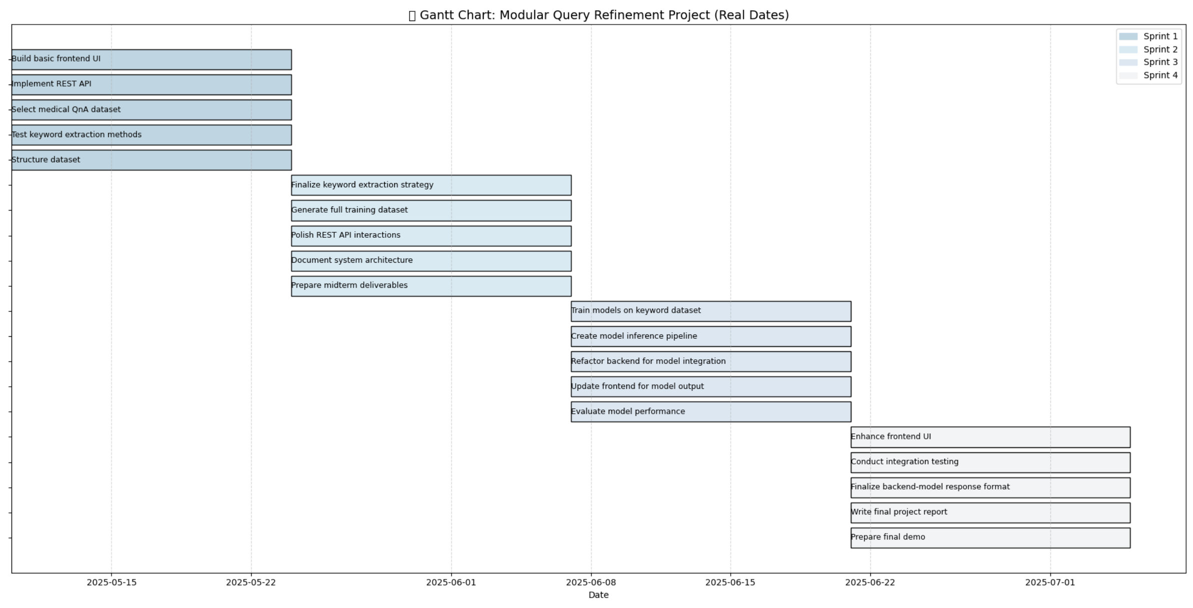


Figure 25: Gantt Chart

12 Conclusion

The application of LDA, LSI, and BERT for topic modeling in the medical domain has proven to be effective in uncovering relevant terms and underlying themes from user queries. Each algorithm brings unique strengths: LDA and LSI offer interpretable topic structures and work well for general exploratory analysis, while BERT demonstrates superior contextual understanding, especially for complex or symptom based queries. Incorporating verb filtering in LDA slightly improved topic focus by reducing irrelevant terms. Overall, the results highlight that while traditional models still hold value in structured environments, transformer-based models like BERT are better suited for capturing semantic nuances and delivering more accurate, user-aligned outputs. This suggests a promising direction for enhancing search systems and clinical decision support tools with deep learning-based language models ‘

13 Futher Works

13.1 Integration of Retrieval-Augmented Generation (RAG):

A promising future enhancement is the integration of Retrieval-Augmented Generation (RAG) to improve both the accuracy and intelligence of query refinement. By combining a retriever (to fetch relevant context) with a generator (to produce refined queries), RAG enables dynamic query rewriting based on real-time information. This approach can enhance domain-specific understanding, support conversational search, and allow seamless updates as knowledge bases evolve. Implementing RAG would enable the system to generate more context aware, relevant, and adaptive query suggestions making it especially valuable for applications like medical search assistants or intelligent help desks.

13.2 Multi-Language Support:

Another interesting and important feature that we can expand in future is to add the support for multi-language. By integrating multilingual models , the system can process user queries in various native languages and still deliver accurate refinements. This would make the application more inclusive and useful in global contexts, especially in regions where English is not the primary language. Additionally, cross-language retrieval capabilities can be introduced to allow users to search

References

[1] M. E. Maron and J. L. Kuhns. “On Relevance, Probabilistic Indexing and Information Retrieval”. In: *J. ACM* 7.3 (1960), pp. 216–244. DOI: 10.1145/321033.321035.

[2] J. J. Rocchio. “Relevance feedback in information retrieval”. In: *The Smart Retrieval System - Experiments in Automatic Document Processing*. Ed. by G. Salton. Englewood Cliffs, NJ: Prentice-Hall, 1971, pp. 313–323.

[3] K. Spärck Jones. “A Statistical Interpretation of Term Specificity and Its Application in Retrieval”. In: *Journal of Documentation* 28.1 (1972).

[4] “Latent Dirichlet Allocation”. In: *Computational Statistics Data Analysis* (2006). DOI: 10.1016/j.csda.2006.11.006.

[5] “Latent Semantic Indexing: Improving Search Results”. In: *Indonesian Journal of Electrical Engineering and Informatics* 12.2 (2023). DOI: 10.11591/eei.v12i2.4602.

[6] J.G. Borade, A.W. Kiwelekar, and L.D. Netak. “Automated grading of PowerPoint presentations using latent semantic analysis”. In: *Revue d’Intelligence Artificielle* 36.2 (2022), pp. 305–311.

14 Appendix

14.1 Query Submission Endpoint

- **Base URL**: `http://127.0.0.1:8000/query/submit/`
- **Method**: POST
- **Description**: Processes natural language queries about medical symptoms and returns relevant keywords using different analysis methods.

Request

Headers:

- `Content-Type: application/json`

Body Parameters:

Parameter	**Type**	**Required**	**Description**
query	string	Yes	The natural language query to analyze
flag	string	Yes	The analysis method to use (see below)

Available Flags:

- `BERT`: Returns a simple list of relevant keywords
- `LSA`: Returns keywords with Latent Semantic Analysis scores
- `UDA_VERB`: Returns keywords with verb-focused analysis scores
- `IDA`: Returns keywords with disease-focused analysis scores

Responses

Success Response (200 OK):

- Returns a JSON object with relevant keywords, formatted differently based on the flag.

Response Examples:

Listing 1: BERT Response

```
{
  "keywords": ["kidney", "urine", "bladder", "blood", "urinary",
               "stone", "cause", "disease", "tract", "dialysis"]
}
```

Listing 2: LSA Response

```
{
  "keywords": [
    ["cancer", -0.29708316497052915],
    ["resource", 0.18949593917846497],
    ["blood", -0.18168937312646596],
    ["management", 0.17981053252474121],
    ["trial", -0.17238994533412967]
  ]
}
```

Error Responses:

- `400 Bad Request`: Invalid or missing parameters
- `500 Internal Server Error`: Server-side processing error

Example Requests

Listing 3: BERT Example

```
curl -X POST http://127.0.0.1:8000/query/submit/ \
  -H "Content-Type: application/json" \
  -d '{"query":"pain in urinary tract", "flag":"BERT"}'
```

14.2 Snapsorts of frontend

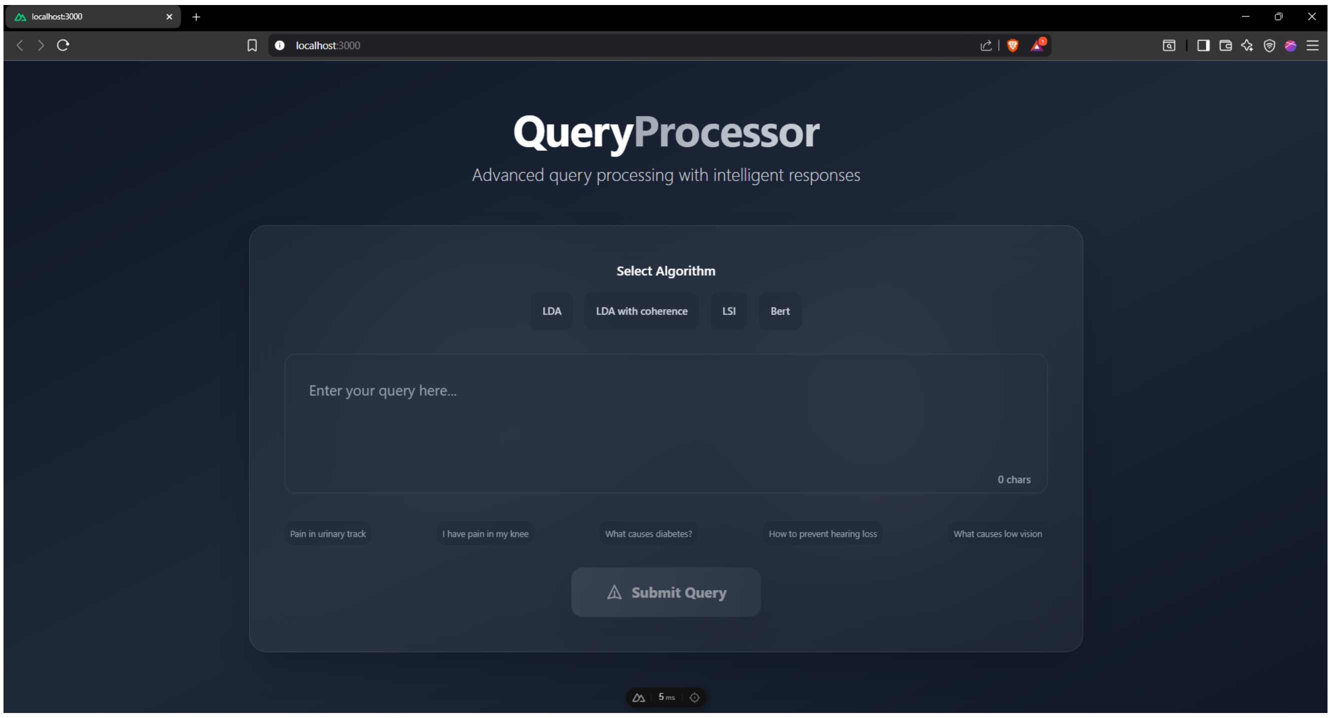


Figure 26: Landing Page

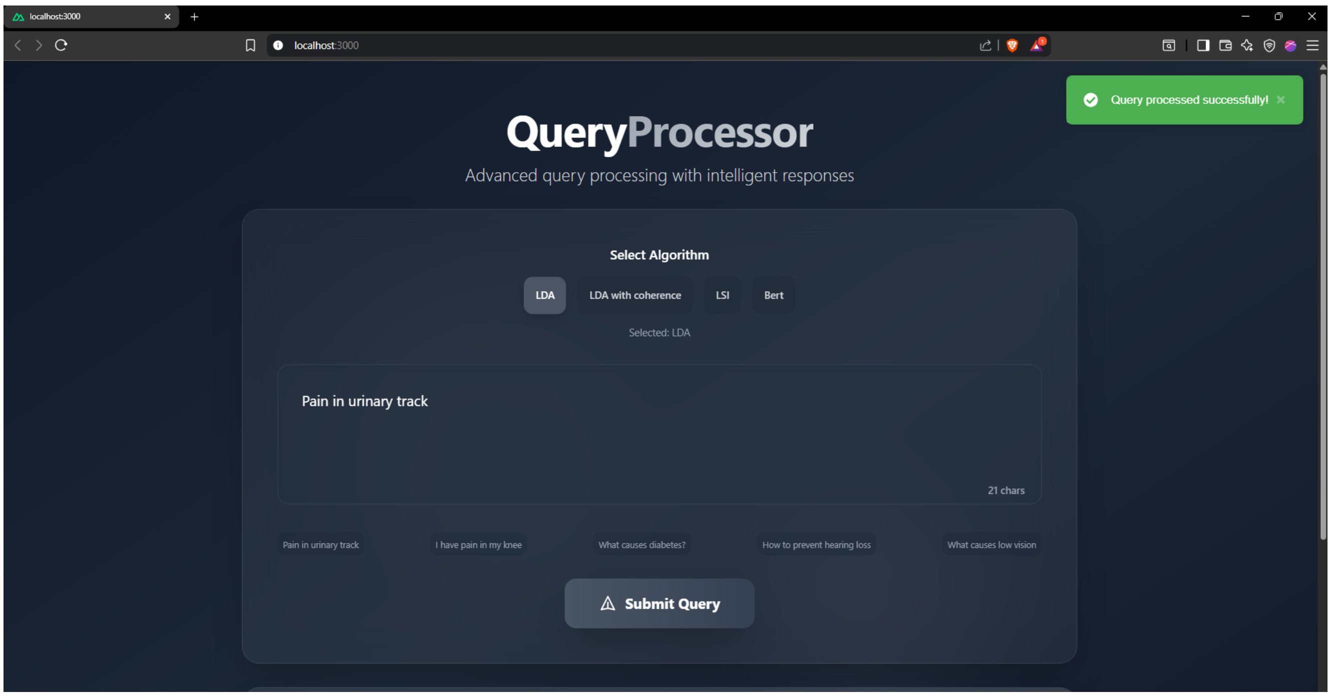
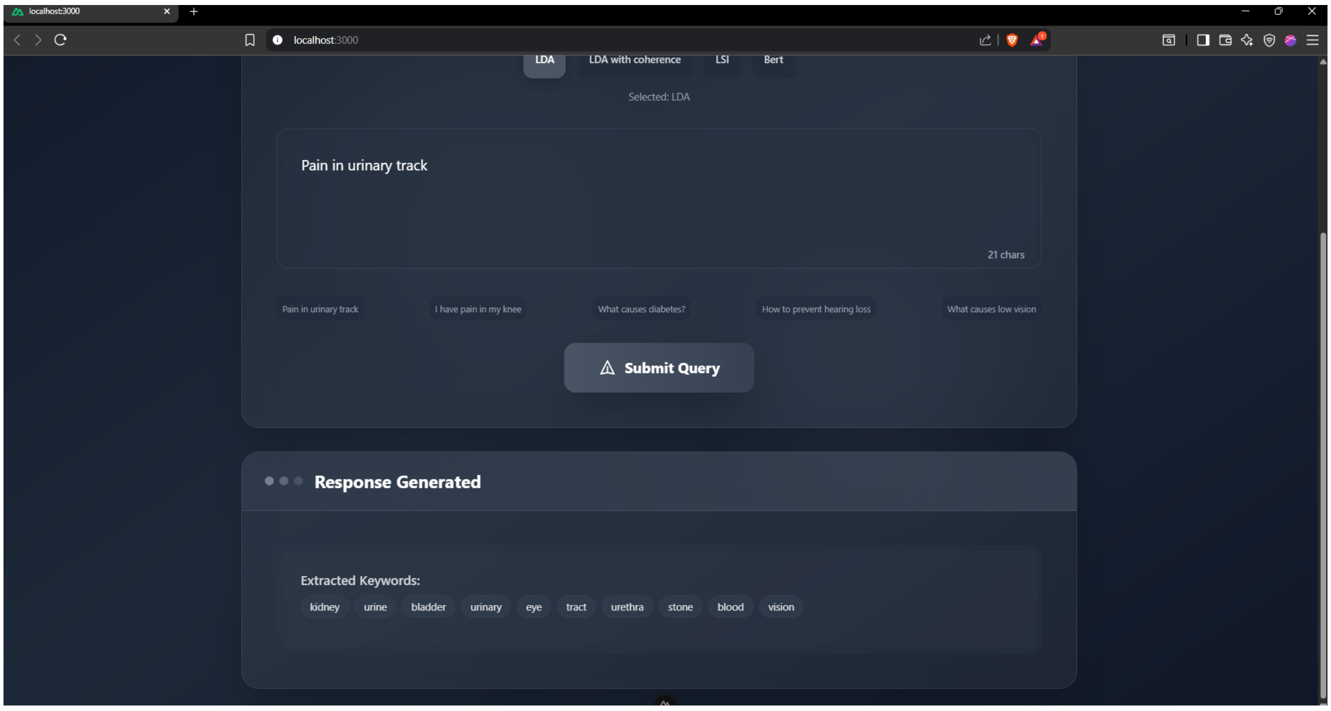


Figure 27: Query Submit



Pain in urinary track

Extracted Keywords:
kidney urine bladder urinary eye tract urethra stone blood vision

Figure 28: Generated Keywords