AI-POWERED SYMPTOM CHECKER

Coordinators:

M. Hakim HAFIDI

M. Hamza GAMOUH

M. Yasser ADERGHAL

Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

الجامعة الدولية للرباط
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵜⴰⴳⵔⴰⵖⵍⴰⵏⵜ ⵏ ⵕⵕⴱⴰⵟ
Université Internationale de Rabat

Final Project Documentation:

1.Title Page:

Project Title: AI-Powered Symptom Checker for Medical Insights

Team Members: Ayoub FAKRAOUI, Imane ENNEYA, Nouhaila BOUNOUAR, Yahya MENKARI

Course/Project Name: Power / Soft skills 1: **Artificial Intelligence**

Institution: International University of Rabat

2. Executive Summary:

Problem Statement:

- Retrieving accurate medical information based on symptoms can be challenging due to the ambiguity of symptoms, fragmented data sources, and the lack of context-aware search mechanisms. Traditional keyword-based systems often fail to provide meaningful insights, leaving users overwhelmed or misinformed.

Solution Overview:

- This project combines Neo4j, Qdrant, and PubMed to deliver symptom-based medical insights. Neo4j facilitates symbolic search by querying structured data stored as a graph, while Qdrant employs semantic search using embeddings generated from the symptoms for contextual understanding. When no relevant results are found locally, the system integrates PubMed as a fallback, retrieving articles based on the user's query, dynamically embedding the articles into Qdrant, and providing insights for future searches. These components are orchestrated through a Flask backend, with AI-generated conversational responses crafted using Google Generative AI (Gemini), ensuring empathetic and structured feedback for users.

3. Problem Definition:

Background:

- Challenges in accessing reliable medical information for users describing symptoms.

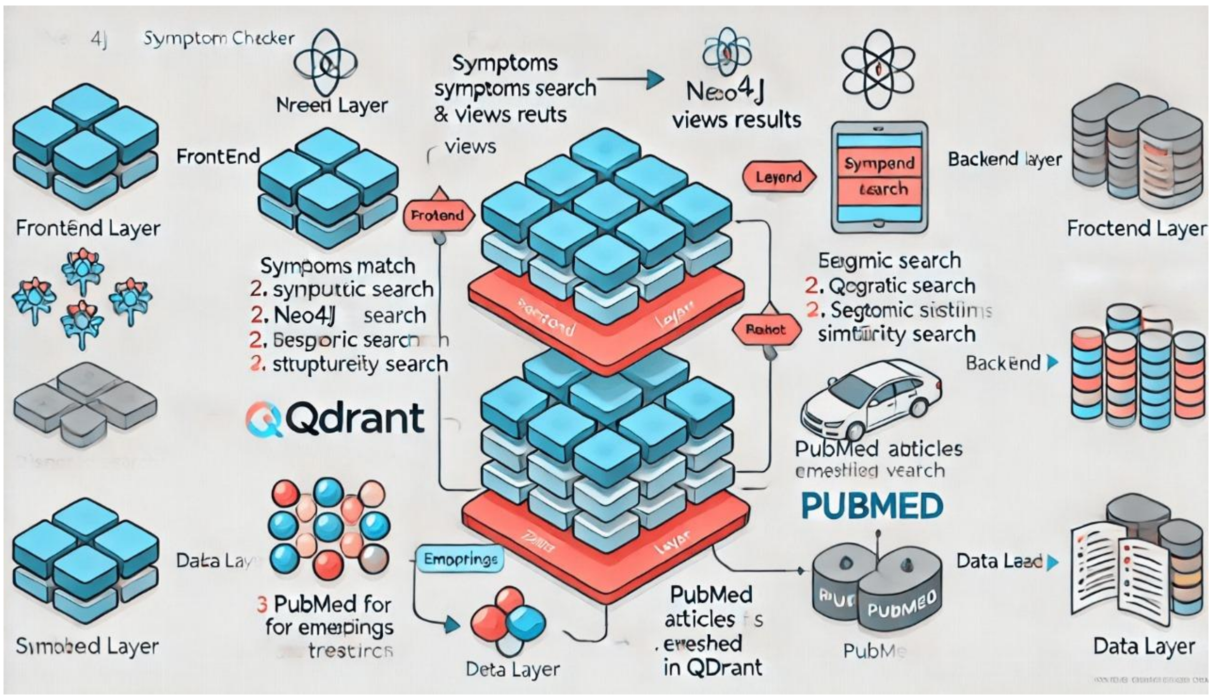
Objectives:

- Enable users to query symptoms and retrieve medical information.
- Seamlessly integrate Neo4j for symbolic search and Qdrant for semantic search.
- Provide fallback to PubMed for additional insights.

4. System Overview:

4.1 System Overview:

- A **diagram** Illustrating the workflow:
  - User Input → Neo4j Query → Qdrant Search → PubMed Fallback →Gemini → Response to User.



- **Tools Used**: Python, Hugging Face, Qdrant, Neo4j, PubMed API, Gemini.

4.2 Workflow:

- Step-by-Step Explanation of the Workflow:

User Input:

- The user inputs symptoms through the user interface.
- Symptoms are extracted using keyword matching or AI-based analysis (extract_symptoms).

Neo4j Query:

The application generates a Cypher query (generate_cypher_query) to search for diseases and treatments in a Neo4j database.
If matches are found, they are returned to the user.

Semantic Search in Qdrant:

If Neo4j does not return results, the application generates an embedding for the symptoms using a pre-trained model (generate_embedding) and searches for similar vectors in the Qdrant collection.
PubMed Integration:

If Qdrant also fails to provide relevant results, the system fetches medical articles from PubMed based on the user's input.
These articles are preprocessed (preprocess_articles), embedded, and stored in Qdrant for future use.

Result Combination and Response Generation:

The results from Neo4j, Qdrant, and PubMed are combined and processed.
A generative AI model (e.g., Gemini) summarizes the findings into a user-friendly response, emphasizing the importance of consulting a healthcare professional.
User Feedback:

The results and AI-generated summaries are displayed to the user, completing the interaction.

5. Design and Implementation:

5.1 Backend:

- **Neo4j Integration**:

The Neo4j graph database organizes medical data into four primary node types: **Symptom**, **Disease**, **Medicine**, and **Doctor**. Each node contains properties such as id, name, and additional relevant attributes (e.g., description for diseases and medicines). **Relationships** between nodes include: HAS_SYMPTOM (Disease → Symptom), :TREATED_BY (Disease → Medicine), and :CONSULTED_BY (Disease → Doctor).

- **Qdrant Integration**:

➢ Collection setup and embedding storage.
➢ Semantic search implementation with embeddings.

- **PubMed Data**:

➢ Query construction and metadata extraction.
➢ Preprocessing articles for storage and retrieval.

5.2 AI and Models:

Models Used:

This project leverages two key AI models to enable effective semantic search and context-aware embedding generation:

- **Gemini-1.5-Flash** (from Google Generative AI):

  ➢ Used to process user inputs, generate Cypher queries, and provide conversational responses to enhance the user experience. This model allows the system to interpret symptom descriptions and formulate structured database queries.

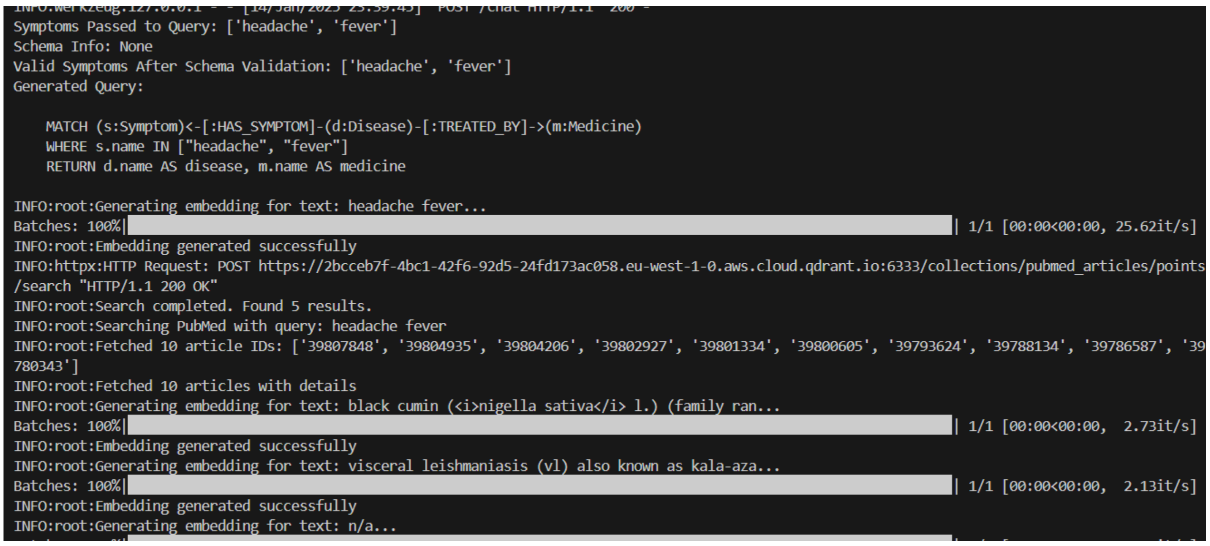
- **Cambridge/SapBERT-from-PubMedBERT-fulltext** (from Hugging Face sentence-transformers):

  ➢ Utilized for embedding generation, enabling semantic search in Qdrant. This model is fine-tuned for biomedical text and provides high-quality vector representations for medical data, ensuring precise and contextually relevant results.

6. Results and Testing:

Visuals:

- Include screenshots of console outputs and search results.

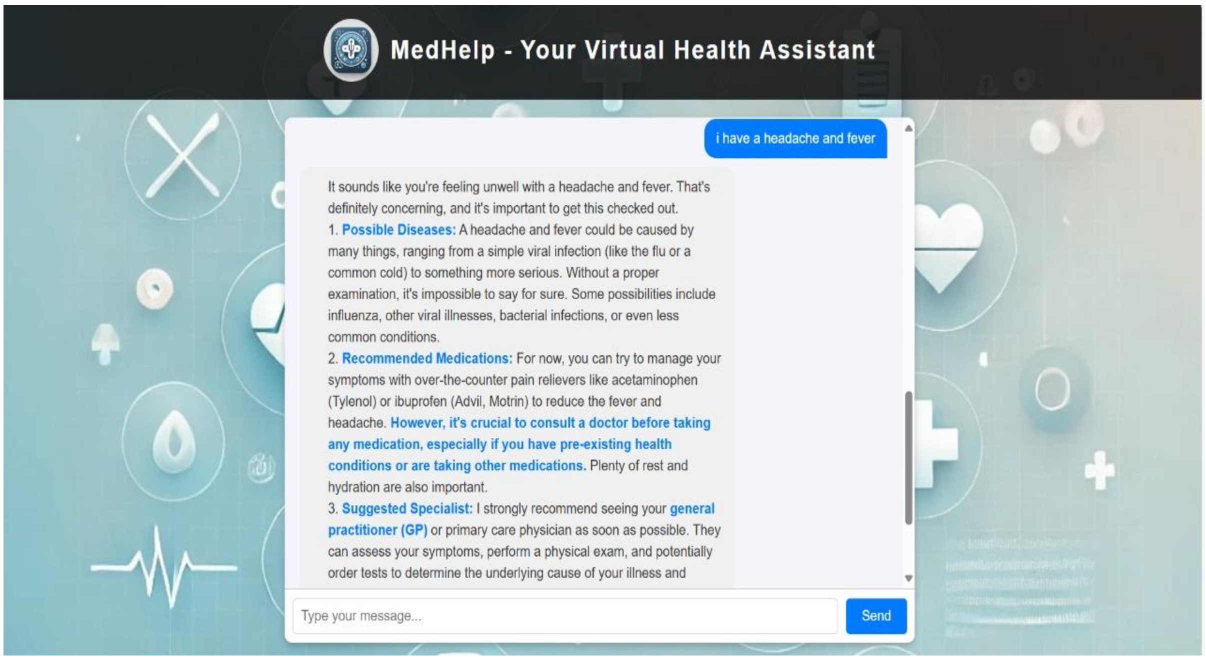


```
INFO:werkzeug:127.0.0.1 - - [14/Jan/2025 23:39:43] "POST /chat HTTP/1.1" 200 -
Symptoms Passed to Query: ['headache', 'fever']
Schema Info: None
Valid Symptoms After Schema Validation: ['headache', 'fever']
Generated Query:

    MATCH (s:Symptom)<-[:HAS_SYMPTOM]-(d:Disease)-[:TREATED_BY]->(m:Medicine)
    WHERE s.name IN ["headache", "fever"]
    RETURN d.name AS disease, m.name AS medicine

INFO:root:Generating embedding for text: headache fever...
Batches: 100%|██████████████████████████████████████████████████| 1/1 [00:00<00:00, 25.62it/s]
INFO:root:Embedding generated successfully
INFO:httpx:HTTP Request: POST https://2bcceb7f-4bc1-42f6-92d5-24fd173ac058.eu-west-1-0.aws.cloud.qdrant.io:6333/collections/pubmed_articles/points
/search "HTTP/1.1 200 OK"
INFO:root:Search completed. Found 5 results.
INFO:root:Searching PubMed with query: headache fever
INFO:root:Fetched 10 article IDs: ['39807848', '39804935', '39804206', '39802927', '39801334', '39800605', '39793624', '39788134', '39786587', '39
780343']
INFO:root:Fetched 10 articles with details
INFO:root:Generating embedding for text: black cumin (<i>nigella sativa</i> l.) (family ran...
Batches: 100%|██████████████████████████████████████████████████| 1/1 [00:00<00:00,  2.73it/s]
INFO:root:Embedding generated successfully
INFO:root:Generating embedding for text: visceral leishmaniasis (vl) also known as kala-aza...
Batches: 100%|██████████████████████████████████████████████████| 1/1 [00:00<00:00,  2.13it/s]
INFO:root:Embedding generated successfully
INFO:root:Generating embedding for text: n/a...
```

- User output:



MedHelp - Your Virtual Health Assistant

i have a headache and fever

It sounds like you're feeling unwell with a headache and fever. That's definitely concerning, and it's important to get this checked out.
1. **Possible Diseases:** A headache and fever could be caused by many things, ranging from a simple viral infection (like the flu or a common cold) to something more serious. Without a proper examination, it's impossible to say for sure. Some possibilities include influenza, other viral illnesses, bacterial infections, or even less common conditions.
2. **Recommended Medications:** For now, you can try to manage your symptoms with over-the-counter pain relievers like acetaminophen (Tylenol) or ibuprofen (Advil, Motrin) to reduce the fever and headache. **However, it's crucial to consult a doctor before taking any medication, especially if you have pre-existing health conditions or are taking other medications.** Plenty of rest and hydration are also important.
3. **Suggested Specialist:** I strongly recommend seeing your **general practitioner (GP)** or primary care physician as soon as possible. They can assess your symptoms, perform a physical exam, and potentially order tests to determine the underlying cause of your illness and

7. Challenges and Solutions:

- **Qdrant Initialization**:
  During the Qdrant initialization, an issue was encountered with handling empty collections. When attempting to perform semantic searches or insert data into a non-existent or uninitialized collection, the system would fail due to the absence of the target collection. To address this, a check was implemented during initialization to verify if the specified collection exists. If the collection was missing, it was automatically created with the appropriate configuration, including vector size and distance metrics. This proactive approach ensured seamless integration of Qdrant into the workflow and prevented runtime errors caused by uninitialized collections, thereby enhancing the system's robustness and reliability.

- **Library Compatibility**:

  During development, a major challenge was ensuring compatibility between the sentence-transformers library and its dependency, transformers. Initially, the project used incompatible versions of these libraries, resulting in errors such as missing functions (is_nltk_available) in transformers.

- **Solution**:

  Identified the compatibility issue by reviewing library documentation and error logs.
  Downgraded transformers to a version compatible with the existing sentence-transformers version.

- **Fallback Logic**:

  When Neo4j and Qdrant failed to provide results, a fallback mechanism was implemented using PubMed. The system queries PubMed via its API to fetch articles related to the symptoms. These articles are preprocessed, and embeddings are generated using the SapBERT model. The embeddings and metadata are then stored in Qdrant, dynamically expanding its database. This ensures users receive relevant results while enhancing the system's knowledge base over time.

8. Future Enhancements

Improvements:

- **Model Fine-Tuning**:

  - Fine-tune the embedding model (SapBERT) on domain-specific medical data to improve semantic search accuracy and relevance.
  - Incorporate a generative AI model, Gemini-1.5-Flash, trained on medical conversations for more context-aware user responses.

- **Improving Qdrant Indexing**:
  - Optimize Qdrant indexing techniques to enhance search speed and scalability for large vector datasets.
  - Experiment with different similarity measures (e.g., Euclidean distance or hybrid approaches) to improve search precision.

- **Adding a GUI**:
  - Design a user-friendly graphical interface for easier symptom input and result visualization.
  - Include features like query history, interactive charts, and detailed article links for PubMed results.

9. Conclusion

This project successfully combines symbolic and semantic search methodologies to provide accurate and contextually rich medical insights. By integrating Neo4j, Qdrant, and PubMed, the system bridges the gap between user queries and reliable medical information.

Key achievements include:

- Accurate symptom-to-disease mapping via Neo4j symbolic search.
- Context-aware results using AI-generated embeddings in Qdrant.
- A dynamic fallback mechanism that ensures users always receive meaningful responses.

The innovative use of AI models and graph-based databases demonstrates the potential for scalable and intelligent medical applications. This system lays a strong foundation for future improvements in accessibility and precision in medical data retrieval.

10. Appendices

Code Repositories:

- GitHUB: https://github.com/ayuuoob/AI-powered-symptom-checker.git