**Final Project Report: Medical Chatbot Using Neo4j and GPT-2 ( Rahul Tammalla , Clarkson University)**

**1. Introduction**

In the field of healthcare, accurate and reliable medical information is essential. However, many AI-powered chatbots suffer from **hallucination**, generating misleading or false information. The goal of this project was to develop a **medical chatbot** that leverages **Neo4j (a knowledge graph) for fact-based retrieval** and **GPT-2 (a language model) for answering general queries**.

The chatbot ensures that when a user asks a question about a medical condition that is **stored in Neo4j, the response is 100% fact-based with zero hallucination**. For queries that are **not covered in Neo4j, GPT-2 generates a response**, but with an **expected 40-50% hallucination rate**.

This project also aimed to **demonstrate the effectiveness of Retrieval-Augmented Generation (RAG) techniques**, where a **knowledge graph is queried before falling back to an LLM**, ensuring improved accuracy and reliability in medical responses.

**2. Objectives**

The key objectives of the project were as follows:

1. **Develop a hybrid chatbot** that combines **structured data retrieval (Neo4j)** with **generative AI (GPT-2)** for answering medical queries.
2. **Implement a knowledge graph using Neo4j** to store structured medical data such as diseases and symptoms.
3. **Minimize hallucination** by prioritizing Neo4j data over GPT-2 for fact-based responses.
4. **Use Natural Language Processing (NLP)** techniques to dynamically extract disease names from user queries.
5. **Deploy the chatbot as a web application** using **Streamlit Cloud** for easy accessibility.
6. **Handle real-world challenges** in AI-based chatbot deployment, including **dependency management, security, and performance optimization**.

**3. Methodology**

The chatbot was developed using a structured pipeline to handle user queries efficiently. The step-by-step workflow is as follows:

1. **User Input**: The user enters a medical question in the chatbot’s interface.
2. **NLP Processing**: The query is processed using **spaCy**, which extracts potential **disease names** from the input.
3. **Querying Neo4j**: The chatbot checks if the extracted disease is present in the **Neo4j knowledge graph**.
4. **Fact-based Retrieval from Neo4j**:
   * **If the disease is found** in Neo4j, the chatbot retrieves its **symptoms and relevant details** from the knowledge graph and returns a **100% accurate response**.
   * **If the disease is not found**, the chatbot **falls back to GPT-2**, which generates a response using its pre-trained model.
5. **Displaying the Response**: The chatbot returns the final answer to the user, clearly distinguishing between a **Neo4j fact-based response** and an **AI-generated response with possible hallucination**.

This **hybrid approach** ensures that **fact-based medical knowledge is prioritized**, while still allowing AI-generated responses for broader queries.

**4. Technology Stack**

The following technologies were used to develop the chatbot:

| **Component** | **Technology Used** |
| --- | --- |
| **Knowledge Graph** | Neo4j |
| **Language Model (LLM)** | GPT-2 (via Hugging Face Transformers) |
| **NLP Processing** | spaCy |
| **Deployment Platform** | Streamlit Cloud |
| **Backend Programming** | Python |
| **Data Storage** | Neo4j Graph Database |

Each of these technologies played a crucial role in ensuring **accuracy, efficiency, and scalability** of the chatbot.

**5. Knowledge Graph and RAG Implementation**

A **knowledge graph** was implemented using **Neo4j** to store structured medical data, including **diseases and symptoms**.

**Knowledge Graph Structure**

The Neo4j database was structured as follows:

* **Nodes**:
  + Disease (e.g., Diabetes, HIV)
  + Symptoms (e.g., Frequent Urination, Weight Loss)
* **Relationships**:
  + (: Disease)- [: HAS\_SYMPTOM]->(: Symptom)

**Retrieval-Augmented Generation (RAG)**

The chatbot implements **Retrieval-Augmented Generation (RAG)** by first **retrieving data from Neo4j** before relying on GPT-2. This ensures that the chatbot:

* **Returns 100% factual responses if data exists in Neo4j.**
* **Uses GPT-2 only when the information is missing from the knowledge graph**, with a clear disclaimer regarding potential hallucination.

This **hybrid RAG approach significantly reduces hallucination**, making the chatbot more reliable for medical queries.

**6. Implementation Process**

**Step 1: Setting Up Neo4j**

* A **Neo4j database** was initialized and populated with medical conditions and their symptoms.
* **Queries were created to fetch diseases and symptoms dynamically**.
* **The database was tested to ensure correct retrieval of medical data.**

**Step 2: Implementing GPT-2 for General Queries**

* **GPT-2 was loaded** from Hugging Face’s **Transformers library**.
* **Response generation was fine-tuned** to reduce hallucination using controlled parameters such as top\_p, temperature, and max\_length.
* **Settings were optimized** to ensure GPT-2 does not generate overly long or duplicate responses.

**Step 3: Integrating NLP for Disease Name Extraction**

* **spaCy was used to process user queries** and extract disease names.
* **Extracted disease names were matched against Neo4j’s database**.
* **If a match was found, Neo4j handled the response**; otherwise, GPT-2 generated an answer.

**Step 4: Testing and Debugging**

* The chatbot was tested extensively with:
  + **Fact-based queries** to ensure Neo4j responded correctly.
  + **Unknown queries** to confirm GPT-2’s ability to generate responses.
  + **Misspelled or rephrased queries** to check NLP accuracy.
* Issues were identified and fixed to **ensure correct disease extraction and knowledge retrieval**.

**Step 5: Deploying Streamlit Cloud**

* A **web interface was created using Streamlit**.
* The chatbot was deployed on **Streamlit Cloud** for public access.

**7. Challenges and Solutions**

**Challenge 1: Neo4j Not Returning Data**

* **Issue**: The chatbot always defaulted to GPT-2, even when Neo4j had data.
* **Solution**: The Neo4j query logic was **fixed to correctly extract disease symptoms**.

**Challenge 2: Missing spaCy Model on Streamlit Cloud**

* **Issue**: en\_core\_web\_sm was missing, causing crashes.
* **Solution**: The model was **added to requirements.txt** for automatic installation.

**Challenge 3: NumPy Version Conflict**

* **Issue**: Deployment failed due to NumPy version mismatches.
* **Solution**: The NumPy version was explicitly set to **ensure compatibility**.

**Challenge 4: Hardcoded Neo4j Credentials**

* **Issue**: Storing database credentials in code posed a security risk.
* **Solution**: Streamlit’s **Secrets Management** was used to store credentials securely.

**8. Deployment and Final Testing**

**Deployment Steps**

1. **Updated requirements.txt** to include all necessary dependencies.
2. **Configured Streamlit Secrets** to store Neo4j credentials securely.
3. **Pushed the code to GitHub** and linked it to Streamlit Cloud.
4. **Restarted the Streamlit app** after fixing all errors.
5. **Conducted final tests** to verify responses from both **Neo4j and GPT-2**.

**9. Conclusion**

This project successfully demonstrated that **combining knowledge graphs with AI models can significantly reduce hallucination in medical chatbots**. By implementing **RAG techniques**, we ensured that fact-based responses were retrieved from Neo4j, while GPT-2 handled general queries when necessary.

**Key Takeaways**

* **Neo4j retrieval ensures 0% hallucination for stored medical data**.
* **GPT-2 serves as a fallback for unknown queries**.
* **NLP dynamically extracts disease names, improving chatbot intelligence**.
* **Streamlit Cloud hosts the chatbot efficiently**.

**10. Future Work**

* **Expand Neo4j's knowledge base** with more diseases and symptoms.
* **Improve GPT-2's medical accuracy** by fine-tuning it on medical datasets.
* **Enable multilingual support** for broader accessibility.

**Output:**

**Neo4j:**

**What are the symptoms of Type 2 Diabetes?**

A screenshot of a computer

AI-generated content may be incorrect.

**What are the symptoms of Asthama?**

A screenshot of a computer

AI-generated content may be incorrect.

**GPT2:**

**What are the symptoms of Heart Attack?**

A screenshot of a computer

AI-generated content may be incorrect.

**What are the symptoms of eye disease?**

**A screenshot of a computer

AI-generated content may be incorrect.**