

MEDICAL CHATBOT FOR BREAST CANCER QUERIES



رواد مصر الرقمية

Prepared by:

- Abdullah Wael
- Abdullah Faisal
- Mohammed Ahmed
- Sarah sherif
- Habiba Ahmed

Presented to:

- Digital Egyptian Pioneers Initiative – DEPI
- Eng/ Mahmoud Radwan

Date: [10/21/2024]

Track: Microsoft Machine Learning

Group code: CAI1_AIS5_S5e

Technical provider: Global Knowledge

Contents

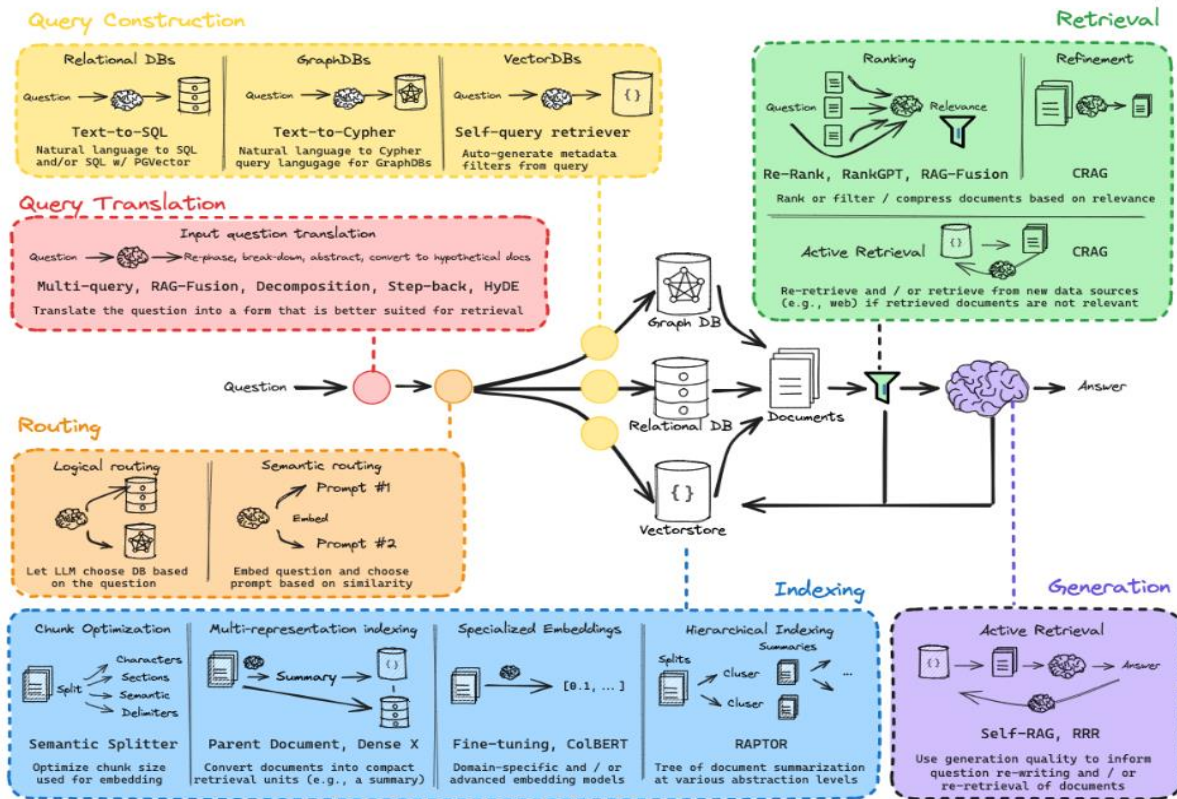
1. Introduction	3
2. Project Overview	3
3. Team Contributions	4
3.1 Data Collection	4
3.2 Model Creation	5
3.3 Deployment	6
4. Code Breakdown	7
4.1 Environment Setup	7
4.2 Model Loading	8
5. User Interaction Experience	10
5.1 User Input	10
5.2 Document Retrieval	10
5.3 Response Generation	10
5.4 Output to User	11
8. Conclusion	11

1. Introduction

This document outlines the development and deployment of a bilingual chatbot designed to assist users with questions about breast cancer. The chatbot uses advanced natural language processing techniques to retrieve relevant information from medical documents and respond to queries in a conversational manner. The project was a collaborative effort, with different team members handling specific components, from data collection to deployment.

2. Project Overview

The chatbot is powered by a combination of state-of-the-art machine learning models, including a transformer-based model for generating responses, and FAISS for efficient document retrieval. This architecture allows the chatbot to quickly search through medical documents and provide accurate answers based on the retrieved context.



3. Team Contributions

3.1 Data Collection

The team was responsible for the crucial task of collecting and preparing the data used in this chatbot. Their primary focus was on ensuring that the data included comprehensive and medically accurate information about breast cancer.

Contributions:

- Data Sourcing:** The team gathered a wide array of medical documents, research papers, and reports related to breast cancer from reputable medical sources.
- Data Cleaning:** They worked on cleaning the data, removing irrelevant or redundant information, to ensure that the content fed into the model was reliable and meaningful.

- **PDF Preparation:** The data included in this project was organized and structured in PDF format for efficient processing.

Click [here](#) to view the data related to breast cancer diagnosis and analysis.

3.2 Model Creation

The team worked collaboratively on creating the core of the chatbot by developing the model responsible for answering the questions. They selected appropriate machine learning models capable of handling medical information retrieval effectively.

Key Contributions:

- **Model Selection:** The team chose the LLaMA 3.1-8B Instruct model for this project, a powerful transformer-based model known for its language generation capabilities. The model was fine-tuned to respond to medical queries in both English and Arabic.
- **Embedding Generation:** They used Sentence-Transformers, a highly effective tool for generating embeddings, which are numerical representations of text that enable the system to understand and retrieve relevant information based on user queries.
- **FAISS Integration:** To enable fast and scalable document retrieval, the team integrated FAISS (Facebook AI Similarity Search), a library that performs efficient similarity searches across documents. This allows the chatbot to sift through thousands of pages of medical literature within milliseconds and retrieve the most relevant content.

```
from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig
```

```
model_id = 'meta-llama/Llama-3.1-8B-Instruct'
tokenizer = AutoTokenizer.from_pretrained(model_id)
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type='nf4',
    bnb_4bit_compute_dtype=torch.bfloat16
)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    quantization_config=bnb_config,
    low_cpu_mem_usage=True
)
return tokenizer, model
```

```
embedder = SentenceTransformer('distiluse-base-multilingual-cased-v2')
pdf_embeddings = embedder.encode([doc.page_content for doc in documents])
```

3.3 Deployment

The team focused on making the chatbot accessible through a web interface, ensuring smooth deployment and integration of all components.

Key Contributions:

- **Deployment Environment:** The team utilized Streamlit to build an intuitive and interactive user interface, making it easy for users to interact with the chatbot.

- **Localtunnel Integration:** They used localtunnel to expose the app to the web, allowing external access through a unique URL.
- **Hugging Face Integration:** The team also worked on integrating the model into the Hugging Face Hub, ensuring secure and fast access to the pretrained models. They managed tokenized access seamlessly using the Hugging Face API.

```
!streamlit run /kaggle/working/app.py & npx localtunnel --port 8501
```

```
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.
```

```
You can now view your Streamlit app in your browser.
```

```
Local URL: http://localhost:8501
```

```
Network URL: http://172.19.2.2:8501
```

```
External URL: http://35.237.2.123:8501
```

```
your url is: https://petite-squids-drive.local.lt
```

4. Code Breakdown

4.1 Environment Setup

The first step in the project implementation was setting up the required environment by installing the necessary libraries. The following libraries were installed:

- Streamlit: A framework used to create the user interface for the chatbot.
- Langchain & langchain-community: These libraries helped in document processing, splitting the content into manageable chunks for embedding generation.
- Sentence-Transformers: This library was used to generate embeddings from the text, converting large documents into vector representations that the model could search through.

- FAISS: A similarity search library used to quickly retrieve relevant sections of the documents based on the user's query.
- Transformers: This library was used to load the pretrained language model, LLaMA, from Hugging Face for natural language generation.
- Hugging Face Hub: To access the LLaMA model and manage tokens securely for model authentication.
- PyPDF: For reading and extracting text from the PDFs that contain the medical information.
- BitsAndBytes: For optimizing the transformer model to work in reduced precision (4-bit quantization), enabling it to run efficiently with lower memory usage.

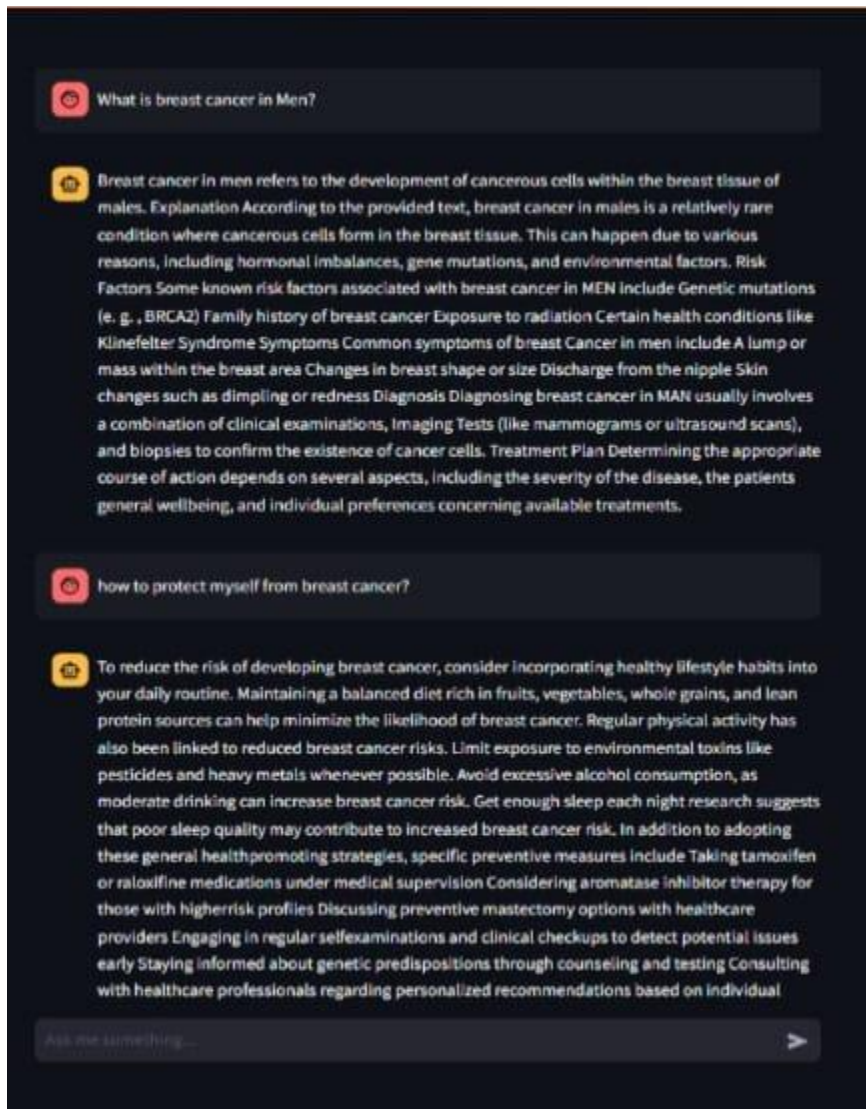
```
Collecting opentelemetry-instrumentation-asgi==0.48b0 (from opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
  Downloading opentelemetry_instrumentation_asgi-0.48b0-py3-none-any.whl.metadata (2.0 kB)
Collecting opentelemetry-instrumentation==0.48b0 (from opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
  Downloading opentelemetry_instrumentation-0.48b0-py3-none-any.whl.metadata (6.1 kB)
Collecting opentelemetry-semantic-conventions==0.48b0 (from opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
  Downloading opentelemetry_semantic_conventions-0.48b0-py3-none-any.whl.metadata (2.4 kB)
Collecting opentelemetry-util-http==0.48b0 (from opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
  Downloading opentelemetry_util_http-0.48b0-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: setuptools>=16.0 in /opt/conda/lib/python3.10/site-packages (from opentelemetry-instrumentation==0.48b0->opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
Requirement already satisfied: wrapt<2.0.0,>=1.0.0 in /opt/conda/lib/python3.10/site-packages (from opentelemetry-instrumentation==0.48b0->opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
Collecting asgiref==3.0 (from opentelemetry-instrumentation-asgi==0.48b0->opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting opentelemetry-api==1.2.0 (from chromadb)
  Downloading opentelemetry_api-1.27.0-py3-none-any.whl.metadata (1.4 kB)
INFO: pip is looking at multiple versions of opentelemetry-sdk to determine which version is compatible with other requirements. This could take a while.
Collecting opentelemetry-instrumentation-fastapi==0.41b0 (from chromadb)
  Downloading opentelemetry_instrumentation_fastapi-0.47b0-py3-none-any.whl.metadata (2.1 kB)
Collecting opentelemetry-instrumentation-asgi==0.47b0 (from opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
  Downloading opentelemetry_instrumentation_asgi-0.47b0-py3-none-any.whl.metadata (2.0 kB)
Collecting opentelemetry-instrumentation==0.47b0 (from opentelemetry-instrumentation-fastapi==0.41b0->chromadb)
```

4.2 Model Loading

The LLaMA model was loaded and prepared using Hugging Face Transformers. This step involved tokenizing the input text and configuring the model for optimal performance in a real-time setting. The following steps were followed:

- Quantization: The model was loaded using BitsAndBytesConfig in 4-bit precision, significantly reducing memory consumption while maintaining accuracy.
- Model Loading: The LLaMA 3.1-8B model was loaded from Hugging Face using secure tokenized access. This model is known for handling both conversational and context-based question answering tasks efficiently.

- Generation Arguments: Specific generation parameters were set, such as a maximum token limit of 256, no repeated n-grams, and a repetition penalty, to ensure the chatbot generated coherent, concise answers.



5. User Interaction Experience

The user interaction experience is crafted to facilitate easy access to information for individuals seeking answers about breast cancer. The chatbot initiates the conversation and guides users through their queries. Here's a step-by-step breakdown of the interaction:

5.1 User Input

Users can input their questions in either English or Arabic. The chatbot employs natural language processing to understand the intent and context of the queries. This process includes:

- Text Preprocessing: The user's input is cleaned and tokenized to prepare it for processing.
- Language Detection: The system detects the language of the input to route the query to the appropriate language model.

5.2 Document Retrieval

Once the input is processed, the chatbot utilizes FAISS to search for relevant documents. This step includes:

- Embedding Generation: The user query is transformed into an embedding using the Sentence-Transformers library.
- Similarity Search: FAISS performs a similarity search to identify the most relevant documents based on the user's query embedding.

5.3 Response Generation

With the relevant documents retrieved, the chatbot generates a response by:

- Context Extraction: The system extracts context from the selected documents to formulate a coherent answer.
- Response Generation: Using the LLaMA model, the chatbot generates a natural language response that is both informative and relevant.

5.4 Output to User

Finally, the generated response is sent back to the user in a conversational format, ensuring clarity and understanding.

8. Conclusion

The deployment of the bilingual chatbot marks a significant achievement in providing accessible information about breast cancer. Through collaboration and advanced technologies, the project team has created a valuable resource that aids individuals in navigating their questions and concerns related to breast cancer. As a standalone project, it exemplifies the potential of integrating AI into healthcare communication.