**Documentation Report: Medical Q&A Chatbot**

This report details the architecture, components, data flow, and features of the **Medical Question Answering (Q&A) Chatbot**, a sophisticated system built to provide information from a specialized medical knowledge base.

**1. Project Overview**

| Attribute | Detail |
| --- | --- |
| **Title** | **Medical Q&A Chatbot** |
| **Primary Goal** | To answer complex medical and health-related questions accurately using a dedicated knowledge base. |
| **User Interface** | **Streamlit** (Interactive Web Application) |
| **Knowledge Base** | **MedQuAD Dataset** (Loaded from Hugging Face: abachaa/MedQuAD) |
| **Key Functionality** | Hybrid Retrieval (Semantic & Keyword Search), Medical Entity Recognition, Real-time Performance Evaluation. |
| **Disclaimer** | **For informational purposes only; not a substitute for professional medical advice.** |

**2. System Architecture and Component Modules**

The system follows a modular design, where responsibilities are separated into distinct Python scripts orchestrated by the main Streamlit application (app (3).py).

| Module | File | Core Function | Key Technologies |
| --- | --- | --- | --- |
| **Main Application** | app (3).py | **Orchestration & UI**: Sets up the Streamlit interface, handles user input, and displays results, statistics, and visualizations. | Streamlit, Plotly |
| **Data Processor** | data\_processor.py | **Data Ingestion & Preparation**: Loads the MedQuAD dataset, cleans the text, and adds essential metadata (length, complexity, document/question types). | Pandas, Hugging Face datasets |
| **Retrieval Engine** | retrieval.py | **Search Core**: Implements the dual-index search mechanism (Semantic and TF-IDF) to fetch the most relevant Q&A pairs. | Sentence-Transformers, FAISS, Scikit-learn |
| **Entity Recognition** | entity\_recognition.py | **Medical NER**: Identifies and extracts key medical concepts (diseases, symptoms, treatments) from user queries and answers. | **spaCy** (en\_core\_sci\_sm fallback to en\_core\_web\_sm) |
| **Evaluation Suite** | evaluation.py | **Performance Tracking**: Computes and logs system metrics (MRR, MAP, NDCG, Precision, Response Time) for analysis. | NumPy, collections |

**3. Data Processing and Knowledge Base Preparation**

The knowledge base is built by the MedQuADProcessor class in data\_processor.py.

**3.1. Data Flow**

1. **Loading**: The system loads the MedQuAD dataset from the Hugging Face hub.
2. **Cleaning**: The raw Q&A text is processed to remove HTML tags, URLs, and perform standard text normalization.
3. **Metadata Addition**: The data is augmented with analytical columns, including:
   * **Question/Answer Length** (for data analysis).
   * **Complexity Level** (Inferred or labeled for targeted evaluation).
   * **Document Type** and **Question Type** (for performance breakdown).
4. **Persistence**: The final, cleaned, and processed DataFrame is saved to a **pickle file** (processed\_medquad.pkl) for fast future loading, bypassing the need to re-process the large dataset on every run.

**4. Advanced Retrieval Mechanism**

The MedicalRetriever class is the heart of the system, employing a powerful dual-search strategy to ensure high recall and precision.

| Retrieval Method | Core Technology | Index Type | Use Case |
| --- | --- | --- | --- |
| **Semantic Search** | **Sentence-Transformers** (all-MiniLM-L6-v2) + **FAISS** | Dense Vector Index (L2 or IP) | Best for understanding the **meaning and context** of a question, even if keywords are mismatched. |
| **Keyword Search** | **TF-IDF** (Term Frequency-Inverse Document Frequency) | Sparse Vector Matrix | Best for exact **keyword matching** and handling very specific terms that might be missed by general embeddings. |
| **Hybrid Search** | Combination | N/A (Post-Search Merging) | The retriever combines the top results from both methods, performs **re-ranking** (e.g., averaging similarity scores), and returns the final set, balancing context and term-overlap. |

**4.1. Retrieval Workflow**

1. User submits a query.
2. The query is simultaneously vectorized by the Sentence Transformer (for FAISS) and the TF-IDF Vectorizer.
3. Both indices are searched independently.
4. The search method combines the top results and performs re-ranking to provide the most relevant final list of Q&A pairs.

**5. Advanced Features and Evaluation**

**5.1. Medical Named Entity Recognition (NER)**

The MedicalEntityRecognizer module is used to enrich the user experience and potentially refine the search.

* It utilizes the **spaCy** library, specifically attempting to load the specialized medical model (en\_core\_sci\_sm) and falling back to the general English model (en\_core\_web\_sm) if the specialized one is unavailable.
* It also uses **regex patterns** as an enhanced fallback for common medical entities like diseases, symptoms, and treatments.
* Extracted entities are displayed in the UI, highlighting the key medical terms identified in the question and the corresponding answer.

**5.2. Evaluation and Analytics**

The **Statistics** tab in the Streamlit application provides a robust view of system health and performance, computed by the EvaluationMetrics class.

| Metric | Purpose | Source |
| --- | --- | --- |
| **MRR (Mean Reciprocal Rank)** | Measures the quality of the ranking of the first relevant result. | Computed from mock test data (e.g., 0.75). |
| **MAP (Mean Average Precision)** | Measures the overall quality of the ranking across all relevant results. | Computed from mock test data (e.g., 0.68). |
| **NDCG (Normalized Discounted Cumulative Gain)** | Measures ranking quality, giving higher weight to highly relevant documents at the top. | Computed from mock test data (e.g., 0.72). |
| **Response Time** | Tracks average latency for real-time performance monitoring. | Logged in evaluation.py. |
| **Performance by Type** | Displays metrics (e.g., Precision) broken down by the question\_type (e.g., 'symptoms', 'treatment'). | Analyzed for targeted model improvement. |
| **Error Analysis** | Lists common failure patterns and recommendations for improvement. | Mock output for strategic development. |

The results are visualized using **Plotly Express** to generate charts like bar plots of Precision by Question Type, providing actionable insights for development.