PalliSahayak: A Multilingual AI-Powered RAG System for Democratizing Palliative Care Advice in India

Abstract: Access to palliative care advice remains a significant challenge in vast and diverse countries like India. This paper details the architecture and system design of PalliSahayak, a novel Retrieval-Augmented Generation (RAG) system aimed at providing accessible palliative care information based on a trusted corpus of medical standard of care documents and protocols. PalliSahayak features robust document processing, a user-friendly web-based admin interface, and critically, a WhatsApp bot integration with native language support (including Hindi, Bengali, Tamil, and Gujarati) for both text and voice interactions. Key innovations include intelligent dependency management for rapid deployment, comprehensive setup validation, intelligent context fusion for improved answer relevance, and an advanced database health monitoring system with automatic corruption detection and rebuilding capabilities to ensure high availability and data integrity. PalliSahayak endeavors to democratize access to palliative care knowledge for India's diverse population, bridging linguistic and technological gaps.

---

1. Introduction

Palliative care, an essential component of healthcare, focuses on improving the quality of life for patients and families facing serious illnesses. However, in India, with its vast population and diverse linguistic landscape, access to specialized palliative care knowledge is limited, particularly in remote and underserved areas. Traditional information dissemination methods often fall short. This paper presents PalliSahayak, a comprehensive AI-powered RAG system designed to bridge this gap. By leveraging a curated corpus of medical standard of care documents and protocols, PalliSahayak provides reliable palliative care advice. Its integration with WhatsApp, a ubiquitous messaging platform in India, and its support for multiple native languages via speech-to-text (STT) and text-to-speech (TTS) capabilities, aim to make this critical information accessible to a broader audience. PalliSahayak is engineered for robustness, featuring automated setup, validation, and a sophisticated database health management system to ensure continuous and reliable operation.

---

2. PalliSahayak: System Architecture

PalliSahayak is a multi-component platform designed for managing, querying, and interacting with a specialized palliative care knowledge base. The overall architecture, depicted in the README, integrates a core RAG pipeline with user-facing interfaces like a web admin UI and a WhatsApp bot.

The main components include:

- **FastAPI Server**: Forms the backbone of PalliSahayak, handling API requests, serving the admin UI, and managing webhooks for WhatsApp integration.
- **RAG Pipeline (Kotaemon-based)**: The core engine of PalliSahayak, responsible for document processing, indexing, retrieval, and generation of answers.
- **Vector Store (ChromaDB)**: Stores document embeddings for efficient similarity search within PalliSahayak.
- **Admin UI (Gradio)**: A web-based interface for system administrators to manage documents, test queries, monitor PalliSahayak, and manage configurations.
- **WhatsApp Bot**: Facilitates user interaction via text and voice messages on WhatsApp, integrated using the Twilio API.
- **STT Service (Groq)**: Transcribes voice messages from users into text for PalliSahayak, supporting multiple Indian languages.
- **TTS Service (Edge TTS)**: Converts textual answers from PalliSahayak back into speech in the user's preferred language.
- **Ngrok Integration**: Provides a public URL for the local server running PalliSahayak, essential for WhatsApp webhook functionality during development and deployment.

The application initializes these components, sets up necessary directories (data, uploads, logs, cache), and can run with or without ngrok.

---

3. PalliSahayak: Core RAG Pipeline and Document Management

3.1. Document Ingestion and Indexing

The RAG pipeline in PalliSahayak is built upon the Kotaemon framework. It supports various document formats including PDF, DOCX, TXT, HTML, and XLSX. The process involves:

1. **Document Upload**: Administrators can upload documents through the "📁 Upload Documents" tab in the Admin UI. They can optionally add JSON metadata (e.g., `{"category": "palliative_care", "topic": "pain_management"}`) to the documents during upload.
2. **Processing and Chunking**: The `FileIndex` component within Kotaemon processes these files, likely involving text extraction and chunking into manageable pieces for embedding. The `SimpleDocumentProcessor` in the simplified version details text extraction (from PDF, DOCX, TXT, MD) and a simple chunking strategy with overlap.
3. **Embedding**: Text chunks are converted into vector embeddings using models like `BAAI/bge-small-en-v1.5` (configurable via `fastembed` or other providers).
4. **Indexing**: These embeddings and their associated metadata are stored in a vector database (ChromaDB by default) for efficient retrieval within PalliSahayak. The system maintains metadata linking chunks back to their original documents, filenames, chunk indices, total chunks, and page counts.

3.2. Retrieval and Generation

When a user query is received by PalliSahayak:

1. **Query Embedding**: The user's query is embedded using the same embedding model.
2. **Similarity Search**: The vector database is searched to find the most relevant document chunks based on a configurable number of top-k results. A relevance threshold (e.g., distance <= 1.5) is applied to filter results.

3.2.1. Intelligent Context Fusion

To enhance the quality and relevance of the context provided to the LLM by PalliSahayak, an intelligent context fusion mechanism is employed. This process analyzes the semantic similarity (distances) of the retrieved document chunks:

- **Distance Analysis**: If multiple chunks are retrieved, their embedding distances from the query embedding are examined. PalliSahayak calculates statistics like the minimum, maximum, and range of these distances.
- **Fusion Decision**: A predefined fusion threshold (e.g., distance range <= 0.15) determines the strategy.
    - If the distances between the top retrieved chunks are very close (i.e., the distance range is below the threshold), it indicates that multiple chunks are highly and similarly relevant. In this case, these contexts are **fused** and all are passed to the LLM to synthesize a comprehensive answer.
    - If the distances are spread out (i.e., the range exceeds the threshold), it suggests that while one chunk might be highly relevant, others are significantly less so. In this scenario, only the **closest context** (the one with the minimum distance) is selected to provide a focused answer and avoid diluting the context with less relevant information.
- **Output**: This mechanism outputs the filtered (either fused or single best) contexts and a flag indicating whether fusion occurred, guiding the LLM's prompt strategy within PalliSahayak.

3.2.2. Answer Generation and Citation

1. **Context Augmentation**: The filtered and potentially fused chunks form the context that is passed to a Large Language Model (LLM) by PalliSahayak.
2. **Answer Generation**: The LLM (e.g., Groq's `llama-3.1-8b-instant` or `gemma2-9b-it`) generates a response based on the provided context and the user's question. The prompt engineering guides the LLM to synthesize information, potentially from multiple sources if context fusion was active, and to provide citations. The prompt explicitly instructs the LLM to extract relevant information even if the question language differs from the context language.
3. **Source Citation**: PalliSahayak aims to provide source documents, and potentially page numbers or section identifiers, for the information, enhancing transparency and

trustworthiness. The LLM is prompted to include citations in the format: `{retrieved from: [Document Name], [Medical Section], page [number]}`. If the LLM fails to include a citation, one can be added automatically by PalliSahayak based on the metadata of the context(s) used.

The `SimpleReasoning` component from Kotaemon is likely involved in orchestrating this query-response flow in the Kotaemon-based version of PalliSahayak.

---

4. PalliSahayak: User Interaction Modalities

4.1. Web-based Admin Interface

The Admin UI of PalliSahayak, built with Gradio, serves as the central control panel. It provides functionalities for:

- **Document Upload and Indexing**: As described in section 3.1.
- **Query Testing**: Allows administrators to test the RAG pipeline with various queries and review responses and sources directly.
- **Index Statistics**: Displays statistics about the document corpus, such as the number of documents and chunks.
- **Configuration Management**: Provides options to configure LLM and embedding model settings.
- **Database Health Monitoring**: Includes a dedicated tab ("🏥 Database Health") for checking database health, viewing corruption scores, and triggering auto or manual rebuilds.
- **Document Management**: Allows viewing and removing documents from the corpus.

4.2. WhatsApp Bot Integration

A key feature for accessibility in India is PalliSahayak's WhatsApp bot integration, managed by the `EnhancedWhatsAppBot` class using the Twilio API.

4.2.1. Setup

- **Twilio Account**: A Twilio account is used for WhatsApp Business API access. The free tier provides a sandbox environment for testing.
- **API Keys**: `TWILIO_ACCOUNT_SID`, `TWILIO_AUTH_TOKEN`, and `TWILIO_WHATSAPP_FROM` are configured in the `.env` file.
- **Ngrok**: For local development, `ngrok` is used to expose the local server of PalliSahayak to the internet, providing a public URL (e.g., `https://your-ngrok-url.ngrok.io`). This URL is configured as the webhook in the Twilio console.

- **Webhook**: The FastAPI application of PalliSahayak exposes a `/webhook` endpoint that Twilio calls when a message is sent to the bot's WhatsApp number.

4.2.2. Text-based Interaction

Users can send text queries directly to the PalliSahayak WhatsApp bot.

- The `EnhancedWhatsAppBot` processes these incoming messages via the Twilio webhook.
- The text is passed to the RAG pipeline of PalliSahayak for an answer.
- The generated textual answer, including sources, is sent back to the user on WhatsApp by the `TwilioWhatsAppAPI`.

4.2.3. Voice-based Interaction

PalliSahayak supports voice message queries in multiple Indian languages.

- **Voice Input**: Users can send a voice message to the WhatsApp bot.
- **Audio Download**: The bot downloads the audio file using the `MediaUrl0` provided by Twilio, via the `TwilioWhatsAppAPI`.
- **Speech-to-Text (STT)**: The `EnhancedSTTService` uses Groq's Whisper API (`whisper-large-v3` model) to transcribe the audio into text. It can automatically detect the language or perform re-transcription if a specific language is detected from the initial transcript.
- **Query Processing**: The transcribed text is then processed by the RAG pipeline of PalliSahayak.
- **Textual and Audio Response**: The bot sends the textual answer back to the user. Additionally, the `EnhancedTTSService` converts this text answer into an audio response in the detected language, which is also sent to the user.

---

5. PalliSahayak: Multilingual Support for India

Recognizing India's linguistic diversity is crucial for democratizing access. PalliSahayak incorporates features for robust multilingual interaction:

5.1. Speech-to-Text (STT)

The `EnhancedSTTService` within PalliSahayak is responsible for transcribing audio messages.

- **Provider**: Uses Groq's API with the `whisper-large-v3` model.
- **Supported Languages**: Currently supports Hindi, Bengali, Tamil, Gujarati, and English.

- **Language Detection**: It employs pattern matching on the transcribed text to detect the language. For improved accuracy, it may re-transcribe the audio by explicitly specifying the detected language to the Whisper API.
- **Output**: Returns the transcribed text and the detected language code (e.g., 'hi', 'bn').

5.2. Text-to-Speech (TTS)

The `EnhancedTTSService` in PalliSahayak generates audio responses.

- **Provider**: Uses `edge-tts` library, which leverages Microsoft Edge's online TTS services.
- **Supported Voices**: Provides specific neural voices for supported Indian languages (e.g., `hi-IN-SwaraNeural` for Hindi, `bn-IN-TanishaaNeural` for Bengali, `ta-IN-PallaviNeural` for Tamil, `gu-IN-DhwaniNeural` for Gujarati, `en-IN-NeerjaNeural` for English). This is also mentioned as configurable in `config.yaml` in the main README.
- **Functionality**: Takes the textual answer from the RAG pipeline and the target language (usually the language detected from the user's query or their preference) to synthesize an audio file (MP3). This audio file is then made accessible via a public URL for Twilio to send.

5.3. Language Preference Management

- **Automatic Detection**: For voice queries, PalliSahayak automatically detects the language of the input audio and responds in the same language.
- **Manual Selection**: Users can explicitly set their preferred language using commands like `/lang hi` or `/lang bn` via WhatsApp. This preference is stored (e.g., in `user_preferences` dictionary in `EnhancedWhatsAppBot`) and used for subsequent interactions.
- **Default Language**: For text queries where language isn't explicitly set or detectable, a default (e.g., Hindi) might be used by PalliSahayak for TTS responses.

---

6. PalliSahayak: System Robustness: Database Health and Auto-Rebuild

To ensure reliability, especially when documents are frequently added or removed, PalliSahayak incorporates an advanced Database Health Monitoring and Auto-Rebuild feature. This is crucial for maintaining the integrity of the vector database. The `simple_rag_server.py` provides a detailed implementation of such a manager (`VectorDBHealthManager` and `AutoRebuildManager`).

- **Corruption Detection**:

    - **Multi-layer checks**: PalliSahayak performs checks for database connectivity, metadata synchronization with the vector DB, query functionality, embedding quality, and index integrity.
    - **Health Scoring**: A corruption score can be generated, indicating severity (minor, moderate, critical).
    - **Triggers**: Corruption might be detected after frequent add/remove operations or through routine health checks.
- **Auto-Rebuild System**:
    - **Smart Recovery**: If corruption is detected, PalliSahayak can automatically trigger a rebuild of the vector database.
    - **Backup**: Before rebuilding, a backup of the current state (e.g., metadata) is created to allow for rollback if the rebuild fails.
    - **Re-indexing**: PalliSahayak collects all source documents (from the `uploads/` directory) and re-indexes them by re-processing and re-embedding the files.
    - **Validation**: After rebuilding, the new database is validated to ensure functionality.
    - **Zero-Downtime (Goal)**: The design aims for continued service during rebuilds, though the specifics of achieving zero-downtime are complex.
- **Admin UI Integration**:
    - The "🏥 Database Health" tab in the Admin UI of PalliSahayak allows administrators to:
        - **Check Health**: Manually initiate a scan for corruption issues.
        - **Auto Rebuild**: Trigger the automatic rebuild process, which typically only proceeds if corruption is found.
        - **Force Rebuild**: Manually force a complete rebuild regardless of the current health status.
        - **View Logs**: Monitor the rebuild progress and results.
- **Programmatic Access**: PalliSahayak allows for programmatic health checks and triggering rebuilds via API endpoints (e.g., `/api/health-check`, `/api/rebuild`). Queries can also be designed with an auto-recovery mechanism that checks health and triggers a rebuild if necessary before retrying the query.

This proactive approach to database integrity significantly enhances PalliSahayak's reliability and uptime.

---

7. PalliSahayak: Democratizing Palliative Care Access in India

PalliSahayak holds significant potential for democratizing palliative care advice in India:

- **Accessibility**: By using WhatsApp, a platform with over 500 million users in India, PalliSahayak meets users where they are, requiring no new app downloads or complex interfaces.

- **Linguistic Inclusivity**: Support for major Indian languages (Hindi, Bengali, Tamil, Gujarati) in both text and voice significantly lowers the barrier to access for non-English speakers and those with limited literacy. Voice interaction further simplifies use for individuals uncomfortable with typing or navigating complex UIs.
- **Standardized Information**: Basing answers on a curated corpus of medical standard of care documents ensures that the advice provided by PalliSahayak is reliable and aligned with established medical protocols. This is crucial for maintaining quality and trust.
- **Scalability**: An AI-driven system like PalliSahayak can handle a large volume of queries simultaneously, providing immediate responses without the delays often associated with human-staffed helplines.
- **Cost-Effectiveness**: Leveraging free tiers of services like Groq for LLM/STT and local/open-source solutions for TTS (Edge TTS) and vector storage (ChromaDB) can make PalliSahayak economically viable to operate and scale.
- **Empowerment**: Providing readily available information through PalliSahayak empowers patients, families, and even local healthcare workers in remote areas with knowledge to make informed decisions and manage palliative care needs more effectively.

---

8. Conclusion and Future Directions

PalliSahayak, the AI-powered RAG system presented, offers a robust and accessible solution for disseminating palliative care information in India. Its architecture, combining a sophisticated RAG pipeline with user-friendly WhatsApp integration and vital multilingual support, is tailored to the unique needs of the Indian context. The emphasis on system robustness through database health monitoring, auto-rebuild mechanisms, and intelligent context fusion further ensures its reliability and the quality of information provided.

Future directions for PalliSahayak could include:

- **Expanding Language Support**: Adding more Indian languages and dialects.
- **Personalization**: Allowing for more personalized advice based on (anonymized and consensual) user history or specific patient conditions, while strictly adhering to privacy and ethical guidelines.
- **Integration with Telemedicine**: Creating pathways to connect users with human palliative care specialists for complex cases.
- **Offline Capabilities**: Exploring options for limited functionality in areas with poor internet connectivity.
- **Continuous Corpus Improvement**: Implementing a feedback loop for medical professionals to review and refine the knowledge base and system responses.
- **Clinical Validation**: Conducting studies to assess the accuracy, utility, and impact of the advice provided by PalliSahayak.

By addressing technological and linguistic barriers, PalliSahayak represents a significant step towards making essential palliative care knowledge more equitable and accessible across India