Invisible Disability Assistant

Author: Stephanie Wainaina

Languages: English & Swahili

Target Group: Persons with Invisible Disabilities in Nairobi

Table of Contents

1. Executive Summary

The Invisible Disability Assistant is a bilingual, AI-powered application that supports users with invisible disabilities in navigating Nairobi's public transport. This project leverages two complementary architectures:

- **A Retrieval-Augmented Generation (RAG) System**: Implemented and submitted as a fully working Streamlit app.
- **A Fine-Tuned LLM Extension**: Conceptually developed to explore offline, language-aware enhancements using GPT-2.

Together, these components position the assistant as both an online and offline solution for inclusive and accessible transport assistance.

2. System Architecture Overview

2.1 Submitted System: RAG + ChatGroq

Component	Purpose
Streamlit	Interactive web interface
LangChain	Document retrieval and orchestration
FAISS	Embedding-based vector search
HuggingFace	Text embedding model
ChatGroq (DeepSeek)	LLM for final answer generation
deep-translator	Translation of assistant replies
gettext	Translated UI for English and Swahili

Process:

1. User selects language and submits a query.
2. LangChain retrieves matched documents via FAISS.
3. ChatGroq generates a concise, context-based answer.
4. If needed, the answer is translated and presented.

2.2 Conceptual Extension: Fine-Tuned GPT-2 LLM

Component	Purpose
GPT-2 (fine-tuned)	Offline Q&A using route-specific Q&A training
Q&A Generator	Produces English + Swahili question-answer pairs
HuggingFace Trainer	Fine-tunes GPT-2 using conversational data
Tokenizer + Dataset	Manages preprocessing and batch training

Process:

1. Raw data from Matatu_Routes.csv is transformed into structured prompts.
2. GPT-2 is trained to recognize transport-related queries in both languages.
3. The model answers without retrieval, offering full fluency and offline support.

3. Submitted Implementation Highlights

- **Language-Aware UI**

```python
# Language sidebar selection
language = st.sidebar.selectbox("Select Language", ["English", "Swahili"])

# Configure gettext for UI translations
locales_dir = "locales"
translation = gettext.translation(
    "messages",
    localedir=locales_dir,
    languages=["sw" if language == "Swahili" else "en"],
    fallback=True,
)
translation.install()
_ = translation.gettext  # Shortcut for gettext translations

# Configure AI response translation
translator = GoogleTranslator(
    source="auto", target="sw" if language == "Swahili" else "en"
)
```

- **Cached Retrieval & LLM**

```python
# Cache LLM to prevent reloading on every query
@st.cache_resource
def load_llm():
    return ChatGroq(model="deepseek-r1-distill-llama-70b")


# Cache embeddings and vector store for fast retrieval
@st.cache_resource
def load_vector_store():
    if os.path.exists(FAISS_INDEX_PATH):
        try:
            with open(FAISS_INDEX_PATH, "rb") as f:
                return pickle.load(f).as_retriever()
        except Exception:
            st.warning(_("FAISS index is corrupted. Rebuilding..."))
```

- **Async Retrieval Flow**

```python
# Async function to handle retrieval in the background
async def async_retrieve_answer(query):
    return await asyncio.to_thread(rag_chain.invoke, {"input": query})
```

4. Experimental LLM Extension Workflow

- **Training Data Preparation**
  User: How do I get from GPO to Kenyatta?
  Assistant: You can board Matatu Route 7C from GPO. It stops at Kenyatta National Hospital.

- **Fine-Tuning GPT-2**

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized["train"],
    eval_dataset=tokenized["test"],
    tokenizer=tokenizer,
)
trainer.train()
```

- **Result**

  Fine-tuned model answers Matatu questions in both Swahili and English.

  Works offline, making it ideal for use in rural or low-connectivity areas.

5. Comparative Analysis

Feature	RAG Mode (Submitted)	LLM Mode (Experimental)
Online Dependency	Yes	No
Retrieval-based reasoning	Yes	No
Fluency in freeform Q&A	Moderate	High
Swahili support	Via translation	Native during training
Deployment complexity	Medium	High (model training + tuning)
Response speed	Fast (cached + async)	Fast (once trained)

6. Recommendations

1. **Hybrid Integration:**
    - Use the **RAG system as default** for real-time, retrieval-grounded answers.
    - Offer a **toggle for "LLM Mode"** for more fluent or offline scenarios.
2. **Edge Deployment:**
    - Quantize the fine-tuned GPT-2 model to support deployment on mobile or Raspberry Pi.
3. **Voice-Enabled Queries:**
    - Use Whisper or Google STT for speech-to-text in both languages.
4. **Enhanced Accessibility:**
    - Add visual cues, haptic feedback, and voice confirmation for users with cognitive disabilities.
5. **Community Feedback Loop:**
    - Let users flag inaccurate responses and retrain the LLM incrementally.
6. **Emergency Mode:**
    - Combine SOS triggers with auto-location sharing + predefined AI response.

7. Conclusion

The Invisible Disability Assistant showcases an innovative blend of retrieval-augmented reasoning and fine-tuned language modelling to deliver bilingual, inclusive public transport support. The submitted system is ready for deployment, while the conceptual LLM extension paves the way for resilience, fluency, and offline-first AI capabilities.

With this combined architecture, the assistant can support users **anytime, anywhere**, while centering **accessibility, autonomy, and local language fluency**.