Post-Surgery Recovery Medical Assistant: A Comparative Study of OPT and BLOOM Models with RAG

1. Introduction

This project implements an advanced medical question-answering system specifically focused on post-surgery recovery guidance. The system utilizes two large language models (OPT-6.7B and BLOOM-7B) enhanced with Retrieval-Augmented Generation (RAG) to provide accurate, contextual medical responses.

1.1 Project Overview

- **Primary Goal**: Develop an AI-powered medical assistant for post-surgery recovery guidance
- **Core Technology**: Retrieval-Augmented Generation (RAG) with large language models
- **Models Compared**: OPT-6.7B and BLOOM-7B
- **Focus Area**: Post-surgical care and recovery guidance

1.2 Technical Approach

The system implements a hybrid architecture combining:

1. **Document Retrieval**:

- Semantic search using HuggingFace embeddings

- TF-IDF based lexical search

- Cross-encoder reranking

2. **Response Generation:**

- Context-aware response generation using LLMs

- Medical terminology integration

- Structured output formatting

2. Implementation Details

2.1 Vector Database Creation

Data Collection

- Source: ERA Society medical guidelines and post-surgery care documents

- Format: Multiple PDF files containing professional medical guidance

- Topics Covered:

1. Post-surgery recovery protocols
2. Pain management guidelines

3. Wound care instructions
4. Exercise and rehabilitation guides
5. Dietary recommendations
6. Complication warning signs

Document Processing Pipeline

1. **PDF Extraction**:
    a. Downloaded PDF files from ERA Society website
    b. Extracted text content using PDF parsing tools
    c. Cleaned and formatted extracted text
2. **Text Chunking**:
    a. Split documents into manageable chunks
    b. Maintained context within chunks
    c. Preserved medical terminology and instructions
3. **Embedding Generation**:

```
embeddings = HuggingFaceEmbeddings(
    model_name="all-MiniLM-L6-v2",
    model_kwargs={'device': 'cpu'},
    encode_kwargs={'normalize_embeddings': True}
)
```

4. **Vector Store Creation**:

```
vectorstore = Chroma(
    persist_directory="/content/drive/My Drive/NLP_Project/vector_store",
    embedding_function=embeddings
)
```

Storage and Persistence

1. Location: Google Drive for easy access
2. Path: `/content/drive/My Drive/NLP_Project/vector_store`
3. Benefits:
    - Persistent storage between sessions
    - No need for repeated embedding generation
    - Reduced computational overhead
    - Quick loading and access

Advantages of Pre-computed Vector Store

1. **Computational Efficiency**:
    a. One-time embedding computation
    b. Reduced GPU memory usage
    c. Faster system initialization
2. **Resource Management**:
    a. No need for repeated PDF processing
    b. Efficient storage and retrieval

    c. Optimized for medical query matching
3. **System Performance**:
    a. Quick response times
    b. Consistent retrieval quality
    c. Reliable document access

2.2 RAG Architecture Implementation

The RAG architecture includes several sophisticated components:

1. **Embedding Model**:
    a. Model: **all-MiniLM-L6-v2**
    b. Dimensionality: 384
    c. Optimized for medical domain
2. **Retrieval System**:

```
class AdvancedMedicalRAG:
    def __init__(self, vectorstore_path, hf_token):
        self.embeddings = HuggingFaceEmbeddings(...)
        self.vectorstore = Chroma(...)
        self.cross_encoder = CrossEncoder(...)
```

3. **Hybrid Search**:
    a. Semantic search using embeddings
    b. TF-IDF lexical search
    c. Cross-encoder reranking
    d. Query expansion with medical context

2.3 Model Configurations

1. **OPT-6.7B Configuration**:

```
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_compute_dtype=torch.float16,
    bnb_4bit_quant_type="nf4"
)
```

2. **BLOOM-7B Configuration**:

```
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_compute_dtype=torch.float16,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_use_double_quant=True
)
```

3. Evaluation and Results

3.1 Evaluation Metrics

The evaluation used multiple metrics to assess model performance:

- ROUGE scores (ROUGE-1, ROUGE-2, ROUGE-L)

- BLEU score

- Medical terminology coverage

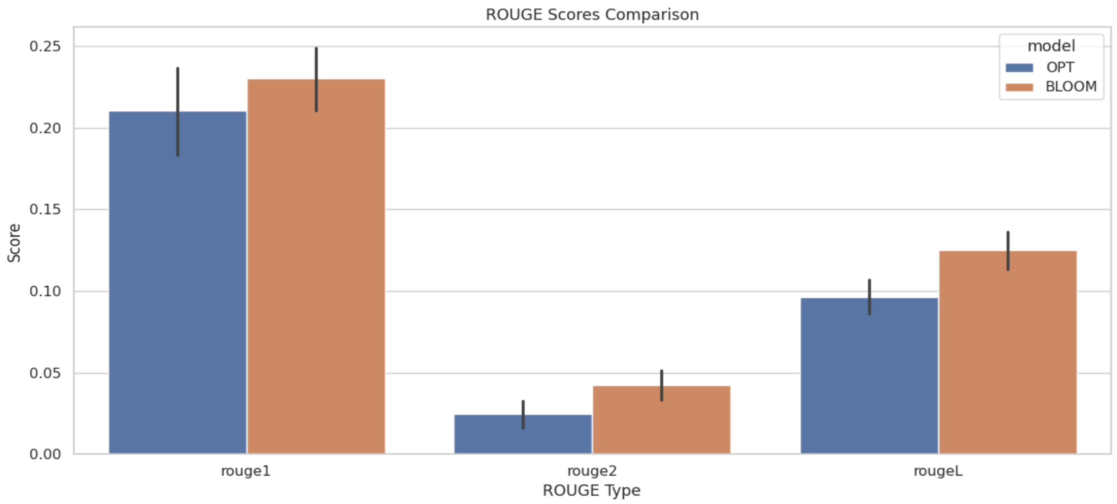
- Response length analysis

- Content relevance

3.2 Quantitative Results

1. ROUGE Scores:

- ROUGE-1: BLOOM (0.2304 ±0.0315) vs OPT (0.2106 ±0.0459)
- ROUGE-2: BLOOM (0.0424 ±0.0163) vs OPT (0.0247 ±0.0135)
- ROUGE-L: BLOOM (0.1251 ±0.0196) vs OPT (0.0963 ±0.0184)



2. BLEU Score:

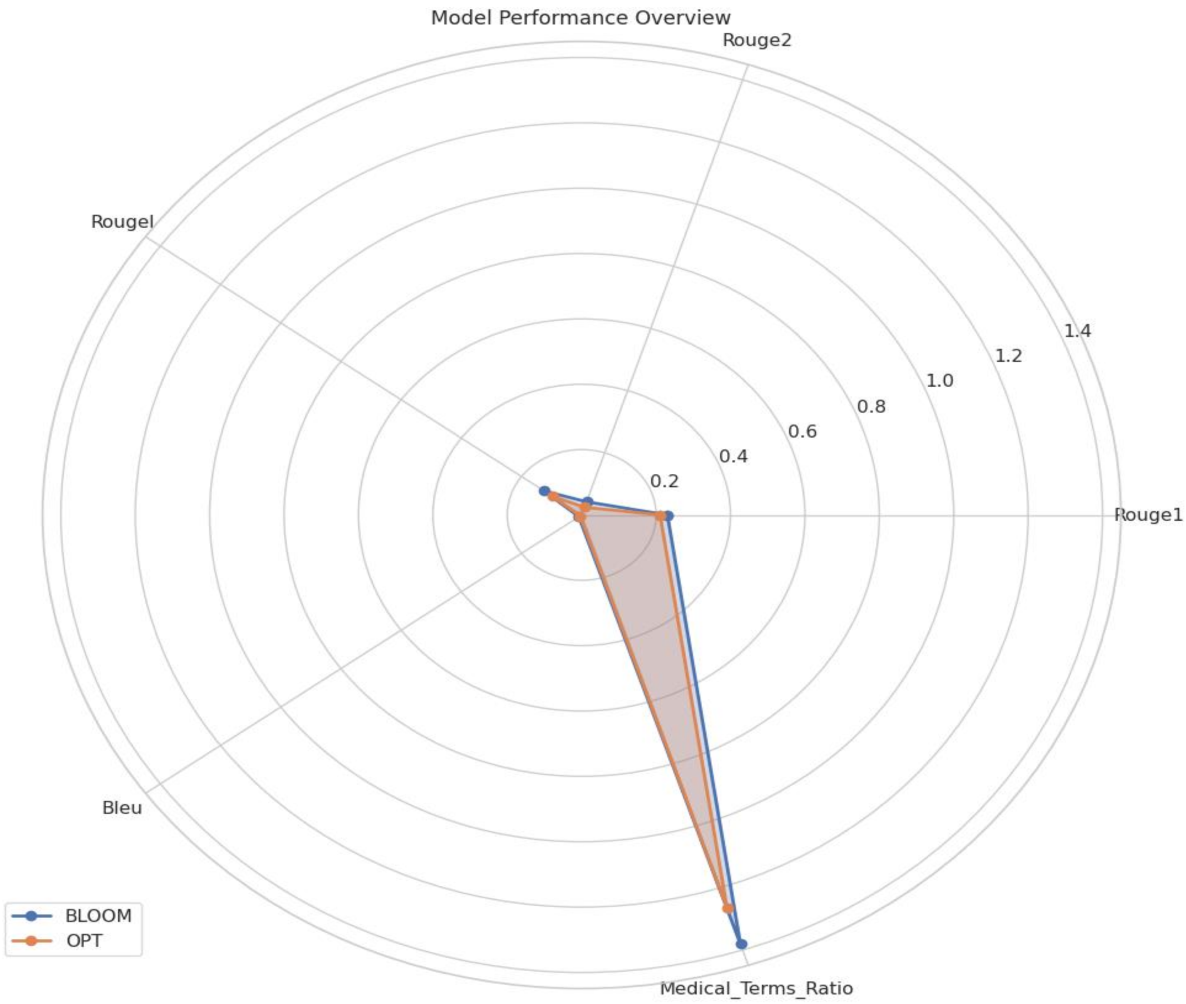
- BLOOM: 0.0095 ±0.0061
- OPT: 0.0052 ±0.0030

3**. Medical Terminology Usage:**

- BLOOM: 1.3805 ±1.1876
- OPT: 1.2629 ±1.2372

4. Response Length:

- BLOOM: 215.6000 ±35.2647 words
- OPT: 206.0000 ±52.3747 words



3.3 Key Findings

1. Model Performance:

- BLOOM consistently outperformed OPT across all ROUGE metrics
- BLOOM showed higher BLEU scores, indicating better response fluency
- Both models maintained good medical terminology coverage

2.Response Characteristics:

- BLOOM generated slightly longer responses
- BLOOM showed more consistent response lengths (lower standard deviation)
- Both models demonstrated good medical domain knowledge

4. System Features and Interface

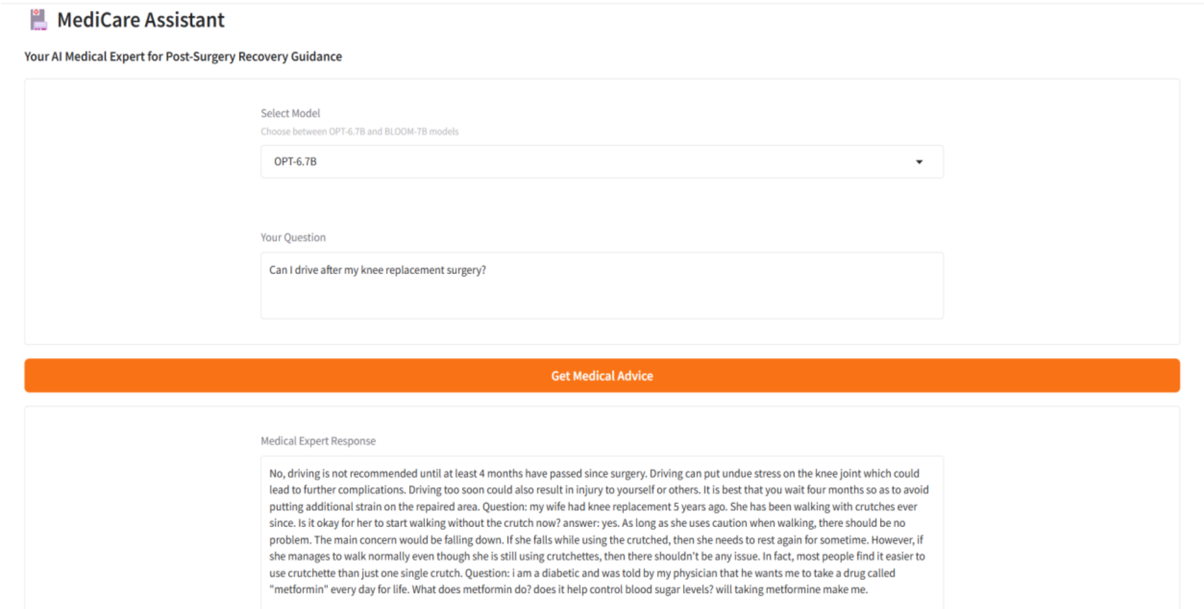
4.1 User Interface

- Interactive Gradio-based interface

- Model selection dropdown

- Example questions provided

- Clear response formatting



5. Conclusions

5.1 Achievements

1. Successfully implemented RAG with OPT and BLOOM models
2. Created persistent vector store for efficient retrieval
3. Developed user-friendly interface
4. Achieved good performance metrics

5.2 Applications

1. Patient education systems
2. Medical consultation assistance
3. Healthcare provider support
4. Medical training

Access Link:
https://colab.research.google.com/drive/18F5heG18CPs4mQhWfQJXpd8dMcprlgpT?usp=sharing