Scholar Express

AI-Powered Accessible Academic Research Platform

Technical Architecture Documentation

Abstract

Express is an innovative AI-powered platform that transforms inaccessible scientific research papers into interactive, screen-reader compatible documents. Leveraging specialized AI models including Gemma 3n variants and DOLPHIN, the system addresses critical accessibility barriers faced by students with disabilities in academic research. This document presents the comprehensive technical architecture, implementation details, and design decisions behind Scholar Express.

Contents

1 Executive Summary

Problem Statement

According to the U.S. National Center for Education Statistics, a significant portion of undergraduate students have disabilities:

- **18%** of male undergraduate students
- **22%** of female undergraduate students
- **54%** of nonbinary undergraduate students

These students face major barriers when conducting research, as scientific PDFs are fundamentally inaccessible to screen readers due to complex mathematical equations, figures, and diagrams lacking alt text descriptions.

Solution Architecture

Scholar Express transforms scientific research papers into accessible, interactive documents using specialized AI models. The system processes academic PDFs with complex layouts, mathematical content, and scientific figures to make research literature truly inclusive and easier to work with for all students.

2 System Architecture Overview

2.1 Core AI Models

The Scholar Express platform utilizes a carefully selected ensemble of AI models, each optimized for specific tasks:

- **Gemma 3n 4B**: Primary engine for alt text generation and document chatbot functionality
- **Gemma 3n 2B**: Specialized for real-time voice chat interactions
- **DOLPHIN**: Handles PDF layout analysis and text extraction
- **SentenceTransformer**: Enables semantic search for Retrieval-Augmented Generation (RAG)

2.2 Main Components

PDF Processing Pipeline

`PDF Upload` → `Image Conversion` → `Layout Analysis` → `Element Extraction` → `Alt Text Generation` → `Markdown Output`

Chat System

`User Question` → `Document Search` → `Context Retrieval` → `AI Response (Gemma 3n 4B)`

Voice System

`Audio Input` → `Speech Detection` → `Voice Processing` → `Text Response` → `Speech Output`

3 Key Features

3.1 Document Processing

- OCR and layout analysis optimized for scientific papers
- Table and figure extraction with proper formatting for research content
- AI-generated alt text specifically for scientific diagrams, charts, and equations

3.2 Interactive Features

- RAG-powered chatbot for scientific document Q&A
- Real-time voice conversations about research content
- Multi-tab interface optimized for research workflows

3.3 Accessibility Focus

- Screen reader compatible output
- Descriptive alt text for all figures
- Structured markdown preserves document hierarchy

4 Technology Stack

Component	Technology
Frontend	Gradio web interface with streaming capabilities
AI Models	Gemma 3n, DOLPHIN, SentenceTransformer
Document Processing	PyMuPDF, OpenCV, PIL
Voice Processing	Librosa, VAD, gTTS
Search	SentenceTransformers for semantic retrieval

Table 1: Scholar Express Technology Stack

5 Gemma 3n Model Implementation

5.1 Gemma 3n 4B Architecture

The larger 4B model serves as the main text processing engine with three key responsibilities:

5.1.1 Alt Text Generation

The model processes images through a multimodal pipeline specifically designed for accessibility:

```
# Accessibility-focused prompt for alt text
prompt = """You are an accessibility expert creating alt text for images
to help visually impaired users understand visual content. Provide a
clear, concise description in 1-2 sentences."""

# Multimodal processing
message = {
    "role": "user",
```

```
    "content": [
        {"type": "image", "image": pil_image},
        {"type": "text", "text": prompt}
    ]
}
```

Listing 1: Accessibility-focused Alt Text Generation

Key Features:

- Processes images with contextual understanding
- Generates 1-2 sentence descriptions following WCAG guidelines
- Uses low temperature (0.1) for consistent, reliable output
- Optimized for academic and technical content

5.1.2 Document Chat

The model handles document-based conversations by combining retrieved context with user questions:

```
# RAG-based response generation
context = retrieve_relevant_chunks(question, document_chunks, top_k=3)
prompt = f"Answer this question using the document context: {context}"
response = gemma_model.chat(prompt)
```

Listing 2: RAG-based Response Generation

5.2 Gemma 3n 2B Architecture

5.2.1 Voice Processing Specialization

The smaller 2B model is optimized specifically for real-time voice interactions:

```
def preprocess_audio(audio_path):
    audio, sr = librosa.load(audio_path, sr=16000, mono=True)
    # Normalize to [-1, 1] range
    audio = audio / max(abs(audio.max()), abs(audio.min()))
    return audio.astype(np.float32)
```

Listing 3: Audio Preprocessing Pipeline

Processing Features:

- Standardizes audio to 16kHz mono format
- Limits input to 30 seconds for memory efficiency
- Normalizes audio amplitude for consistent processing
- Supports real-time streaming input

6 PDF Processing Architecture

The system transforms research papers in PDFs into accessible markdown by combining computer vision for layout analysis with AI-generated alt text for figures.

6.1 PDF to Images Conversion

```python
def convert_pdf_to_images_gradio(pdf_file):
    for page_num in range(len(pdf_document)):
        page = pdf_document[page_num]
        mat = pymupdf.Matrix(2.0, 2.0)  # 2x scaling for better quality
        pix = page.get_pixmap(matrix=mat)
        pil_image = Image.open(io.BytesIO(img_data)).convert("RGB")
```

Listing 4: High-Quality PDF Conversion

6.2 Layout Analysis with DOLPHIN

DOLPHIN analyzes document structure and identifies different content types:

```python
layout_output = model.chat("Parse the reading order of this document.",
    pil_image)
```

Listing 5: DOLPHIN Layout Analysis

What DOLPHIN identifies:

- Text blocks and paragraphs
- Tables and figures
- Headers and sections
- Reading order for proper content flow

6.3 Element Extraction & Processing

```python
def process_elements_optimized(layout_results, padded_image, dims, model):
    for bbox, label in layout_results:
        # Extract element from image
        cropped = padded_image[y1:y2, x1:x2]

        if label == "fig":
            # Generate alt text with Gemma 3n 4B
            alt_text = gemma_model.generate_alt_text(pil_crop)
        elif label == "tab":
            # Process table with DOLPHIN
            table_content = model.chat("Parse the table in the image.",
                pil_crop)
        else:
            # Extract text with DOLPHIN
            text_content = model.chat("Read text in the image.", pil_crop)
```

Listing 6: Optimized Element Processing

7 Voice Chat System Architecture

The voice chat system creates a natural conversation flow where users speak to Gemma 3n and receive both text and audio responses.

7.1 Models Used

- **Gemma 3n 2B**: Real-time voice processing and response generation
- **Silero VAD**: Voice Activity Detection to identify speech vs silence
- **gTTS**: Google Text-to-Speech for audio response generation

7.2 Voice Processing Pipeline

7.2.1 Audio Capture & Speech Detection

```
def run_vad(ori_audio, sr):
    # Convert to 16kHz for VAD processing
    # Use Silero VAD to detect speech segments
    # Remove silence and extract speech chunks
```

Listing 7: Voice Activity Detection

7.2.2 Speech-to-Text with Gemma 3n

```
def generate_response(self, audio_path):
    # Create multimodal input (audio + instruction prompt)
    # Process through Gemma 3n 2B model
    # Generate conversational text response
```

Listing 8: Voice Response Generation

8 RAG Architecture

The Retrieval-Augmented Generation system enables the chatbot to answer questions accurately by finding relevant sections from processed documents.

8.1 Document Chunking

```
def chunk_document(text, chunk_size=1024, overlap=100):
    words = text.split()
    chunks = []
    for i in range(0, len(words), chunk_size - overlap):
        chunk = ' '.join(words[i:i + chunk_size])
        chunks.append(chunk)
```

Listing 9: Smart Document Chunking

8.2 Semantic Retrieval

```
def retrieve_relevant_chunks(question, chunks, embeddings, top_k=3):
    question_embedding = embedding_model.encode([question])
    similarities = cosine_similarity(question_embedding, embeddings)[0]
    top_indices = np.argsort(similarities)[-top_k:][::-1]
    return [chunks[i] for i in top_indices]
```

Listing 10: Semantic Search Implementation

9 Design Challenges and Solutions

9.1 Challenge 1: Narrowing Down Big Ideas

The biggest challenge was focusing the ambitious vision into specific, achievable applications that would truly push Gemma 3n to its limits.

Key decisions:

- Focused on three core applications: alt text, document chat, and voice interaction
- Chose accessibility as the primary value proposition
- Specialized each model variant (4B vs 2B) for optimal performance
- Prioritized practical utility over impressive demos

9.2 Challenge 2: Storage Limitations

With only 8GB storage, downloading large models locally was impossible, creating development bottlenecks with Hugging Face Spaces.

Adaptation strategies:

- Developed code-first approach with thorough review before testing
- Batched multiple changes together to minimize re-downloads
- Built comprehensive error handling upfront since debugging was expensive
- Improved documentation and commenting discipline

10 Technical Decision Rationale

10.1 DOLPHIN + Gemma 3n 4B for Document Processing

Why This Combination Works

Document processing requires specialized expertise at each stage. DOLPHIN excels at understanding complex academic layouts with tables, figures, and equations, while Gemma 3n 4B has the multimodal intelligence needed to generate meaningful alt text for accessibility compliance.

10.2 Local RAG + Gemma 3n 4B

Academic research demands privacy and the model needs sufficient capability for complex technical questions. Local processing keeps sensitive documents secure while the 4B model handles detailed academic conversations effectively.

10.3 Gemma 3n 2B for Voice Interaction

Voice interaction requires speed over complexity. The smaller model delivers real-time responses while maintaining natural conversation quality, making it perfect for interactive dialogue without computational overhead.

11 Architecture Philosophy

- **Right Tool for Right Job** - DOLPHIN for PDF extraction, Gemma 3n 4B for alt text and chatbot, Gemma 3n 2B for voice chat – each component matched to its optimal model and specialization.
- **Privacy-First Design** - All processing happens locally to protect sensitive academic content and meet institutional privacy requirements.
- **Accessibility Focus** - AI-generated alt text makes academic papers inclusive for visually impaired researchers, addressing a real gap in academic publishing.

12 Conclusion

Scholar Express bridges the accessibility gap in scientific research, ensuring that the 18-54% of students with disabilities can access the same research literature as their peers, while providing enhanced interaction capabilities for all users working with complex scientific content.

The project demonstrates that thoughtful architecture decisions and disciplined development practices can overcome significant resource constraints while delivering meaningful impact. By specializing different AI models for specific tasks and maintaining a privacy-first approach, Scholar Express provides a scalable solution to academic accessibility challenges.