Technical Report: EB-1A RFE Risk Analyzer

Author: Anjolaiya Kabiawu **Date:** August 27, 2025

1. Problem Framing and Approach

The process of applying for an EB-1A "Extraordinary Ability" visa is fraught with subjectivity. A key pain point for legal practitioners and applicants is the risk of receiving a Request for Evidence (RFE), which can cause significant delays and uncertainty. Our approach was to leverage a Large Language Model (LLM) to build a tool that can "think like an adjudicator" to preemptively identify these weaknesses. This project evolved from a simple prompt-based system to a sophisticated, data-driven **Retrieval-Augmented Generation (RAG)** pipeline to ensure the feedback is not only intelligent but also grounded in real-world legal precedent.

2. Architecture and Tool Design

The system is designed with a two-phase architecture: an offline data preprocessing phase to build a knowledge base, and an online analysis phase to evaluate petitions.

Phase 1: Offline Knowledge Base Construction (`build_knowledge_base.py`)

The foundation of the system is a rich knowledge base derived from real-world legal outcomes. This phase is handled by a dedicated script that performs the following steps:

1. **Data Scraping:** It loads the full text of the relevant EB-1A policy and standards from a local text file (`eb1a_policy_manual.txt`).
2. **Chunking & Embedding:** The text is split into smaller, semantically coherent chunks. Each chunk is then converted into a numerical vector (embedding) using an open-source `sentence-transformers` model, which runs locally.
3. **Vector Store Creation:** These embeddings are indexed and stored in a FAISS (Facebook AI Similarity Search) vector store, which is then saved to the local disk. This offline process creates a highly efficient, searchable database of official legal standards.

Phase 2: Online RFE Analysis (`main.py`)

When a user wants to analyze a petition, the main application executes the following RAG-enhanced pipeline:

1. **Petition Ingestion & Intelligent Segmentation:** The user's draft petition (`.docx`, `.pdf`, or `.txt`) is ingested. To handle the high variability of real-world petition formats, a robust two-pass segmentation process was engineered:
   a. **Pass 1 (Structure Recognition):** The first part of the document is sent to an LLM (GPT-4o) with a prompt instructing it to identify the document's unique, high-level section headers.
   b. **Pass 2 (Pattern Extraction & Splitting):** A helper function then extracts the most stable and reliable patterns from these headers (e.g., section numbers like "1.1" or "2.4.1"). A specific, dynamic regular expression is built from these patterns and used to split the entire document into precise, meaningful segments. This adaptive approach proved far more effective than static keyword matching.
2. **Initial Analysis (LLM Call #1):** Each petition segment is sent to the GPT-4o API. The prompt instructs the model to act as a USCIS adjudicator, identify potential weaknesses based on the 10 EB-1A criteria, and format the findings in a structured table.
3. **Retrieval-Augmented Generation (RAG):** For each weakness identified, the description of that weakness is used as a query to the FAISS vector store. The system retrieves the most relevant passages from the USCIS Policy Manual.
4. **Enhanced Generation (LLM Call #2):** The original weakness, along with the retrieved legal context, is passed to the GPT-4o API with a new prompt. This prompt instructs the model to act as an expert legal assistant and generate a new, evidence-based suggestion that is explicitly grounded in the provided official standards.
5. **Report Generation**: The final, enhanced analysis is compiled into a professional `.docx` report.

3. Edge Cases or Limitations

While the prototype is effective, it has several known limitations:

- **Criterion Misidentification:** The initial analysis model can still occasionally misclassify a section's criterion due to semantic overlap.

- **Complex Document Formatting:** While the final segmentation parser is robust, extremely complex layouts (e.g., multi-column text, dense tables) could still pose a challenge.
- **Knowledge Base Scope:** The current knowledge base is built from the policy manual. A future version would benefit from including thousands of real-world AAO decisions to understand how policy is applied in practice.
- **API Dependency & Cost:** The system relies on the OpenAI API for its high-level reasoning capabilities, which involves operational costs.

4. Future Extension Opportunities

This prototype serves as a strong foundation for a more robust and feature-rich tool. Key future extensions could include:

- **Intelligent Data Filtering:** The scraping script (`build_knowledge_base.py`) could be enhanced with a classification model to automatically filter out irrelevant (non-EB-1A) cases before they are added to the knowledge base, enabling the use of larger, more diverse data sources.
- **Fine-Tuning a Specialized Model:** The scraped data provides a perfect foundation for fine-tuning a specialized open-source model. This could reduce reliance on the GPT-4o API and create a more cost-effective and expert system.
- **Interactive Web Interface:** The tool could be embedded in a user-friendly web application (using Streamlit or Flask) where an attorney could see the identified weaknesses highlighted directly on their uploaded document.
- **Interactive "Cited Sources”:** The final report could be enhanced to include direct quotes and links to the specific USCIS policy sections that the RAG system used to generate its suggestions, providing ultimate explainability.