Technical Report: EB-1A RFE Risk Analyzer

Author: Anjolaiya Kabiawu **Date:** July 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. An RFE is typically issued when a USCIS adjudicator finds the provided evidence to be ambiguous, unquantified, or insufficient to meet the high standards of the visa category.

The problem is to create a tool that can "think like an adjudicator" to preemptively identify these weaknesses. A simple keyword-based or rule-based system would be too brittle to capture the nuance of legal argumentation.

Our approach was to leverage a Large Language Model (LLM) as a sophisticated reasoning engine. Instead of just performing a technical task (like sentiment analysis), we framed the problem as a role-playing simulation. The core of our solution is a meticulously crafted "master prompt" that instructs the LLM to adopt the persona of a skeptical but fair USCIS officer. This allows the system to move beyond syntax and evaluate the substance of the claims, focusing on generating actionable, human-readable feedback in the format of an internal legal memo.

2. Architecture and Tool Design

The system is designed as a linear pipeline of modular Python scripts, ensuring clarity and ease of maintenance. The architecture prioritizes robust path handling and clear separation of concerns.

1. **Ingestion Module (`document_parser.py`):** The pipeline is initiated from the `main.py` script, which accepts a command-line argument for the input file path. This path is passed directly to the Ingestion Module, which uses the `python-docx`, `PyMuPDF`, and standard `open()` functions to extract raw text from `.docx`, `.pdf`, and `.txt` files, respectively. This direct path handling makes the script flexible and executable from various locations.
2. **Segmentation Module (`document_parser.py`):** Once the text is extracted, a regular expression-based function searches for common structural markers in petitions (e.g., "Criterion 4:", "Personal Statement"). It splits the document into a dictionary of

meaningful chunks. This ensures that the analysis is granular and context-aware, evaluating each criterion's evidence separately.

3. **Analysis Core (ai_analyzer.py):** This is the heart of the system. Each text segment is passed to this module, which wraps it in the master prompt. It then makes an API call to a powerful LLM (e.g., GPT-4o). The prompt instructs the model to identify the relevant EB-1A criterion, look for specific weaknesses (vagueness, lack of quantification), and format its entire output in a structured Markdown format, including a table of findings and the "Adjudicator's Persona Notes."
4. **Report Generation (report_generator.py):** The final module collects the Markdown-formatted analysis from the LLM. To ensure the output is always saved in a predictable location, the script calculates the absolute path to the project's output/ directory. It then parses the Markdown from the LLM and uses the python-docx library to programmatically build a professional Word document.

3. Edge Cases or Limitations

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

- **Criterion Misidentification:** As seen in the generated report, the LLM can occasionally misclassify a section's criterion. This happens when the content of one section (e.g., describing a project for a "Leading Role") is semantically similar to another criterion ("Original Contributions"). The model can prioritize the content's meaning over the section's explicit header.
- **Complex Document Formatting:** The current text extraction and segmentation logic assumes a simple, linear document structure. It would likely fail on documents with complex layouts like multiple columns, embedded tables, or extensive footnotes.
- **Segmentation Brittleness:** The regex-based segmentation is dependent on the petition using clear headings like "Criterion X". If a document uses non-standard phrasing, the segmenter may fail, causing the entire document to be analyzed as one large, un-contextualized block.
- **API Dependency & Cost:** The system's performance is entirely dependent on the availability and capability of a third-party LLM API (e.g., OpenAI). This introduces operational costs per analysis and reliance on an external service.

4. Future Extension Opportunities

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

- **Knowledge Grounding with a Vector Database:** To improve accuracy and mitigate hallucination, the USCIS Policy Manual and a corpus of AAO decisions could be ingested into a vector database (e.g., FAISS or Chroma). The LLM could perform Retrieval-Augmented Generation (RAG), querying this database to ground its analysis in specific legal precedents and provide direct citations in its feedback.
- **Fine-Tuning a Specialized Model:** For cost-efficiency and improved domain-specificity, an open-source model (like Mistral or Llama 3) could be fine-tuned on a proprietary dataset of petitions and their corresponding RFEs. This would create a smaller, faster, and potentially more accurate model for this specific task.
- **Interactive Web Interface:** The tool could be embedded in a user-friendly web application (using Streamlit or Flask). An attorney could upload a document, and the identified weaknesses could be highlighted directly on the text in an interactive side-by-side view, allowing for real-time editing and re-analysis.
- **Bias and Repetition Detection:** The model could be extended to detect more subtle issues, such as a "template-like" feel across different expert support letters by calculating the semantic similarity between them, flagging potential signs of non-genuine authorship.