Payment Transaction Entity Risk Analyser

This Entity Risk Analyser is for financial entities using transaction data to identify the risk with entities involved in the transaction. Data will be ingested from a text file, via Streamlit UI and pass to FastAPI for processing. **XLM-RoBERTa** extracts entity names, which are screened against the OFAC list for compliance risks, assigning a high score if flagged. For unflagged entities, **BART Zeroshot** analyzes news from NewsAPI and also we call SEC EDGAR filings for reputation and anomaly risks and identify anomalies using and give risk and confidence score using **BART Zeroshot**. Then we augment data is fed to an LLM to generate a refined risk score and reasoning for scoring. Results are returned to the Streamlit UI via FastAPI, providing a comprehensive risk assessment.

# Get Input Data

* Purpose: Retrieve transaction data from a text file containing both structured (e.g., "Date: 2023-01-15, Amount: $5000, Entity: ABC Corp") and unstructured (e.g., "ABC Corp paid $5000 on Jan 15, 2023 for services") formats.

## How It Works:

* + Streamlit UI: Use Streamlit, a Python framework, to create a simple web interface where users upload the text file. Streamlit’s st.file\_uploader allows file input, displaying the content for user verification.
  + FastAPI: Implement a backend API using FastAPI to handle file uploads and serve data to the Streamlit frontend. FastAPI processes the uploaded file, reads its contents, and returns the raw transaction data as a JSON response.

## Implementation:

* + Streamlit sends the file to a FastAPI endpoint (e.g., /upload), which reads the text using Python’s open() and read() functions.
  + Store the transactions in memory (**in future in DB**) for further processing.

Tools:

Streamlit ([Streamlit Docs](https://docs.streamlit.io/" \t "_blank)), FastAPI ([FastAPI Docs](https://fastapi.tiangolo.com/" \t "_blank)).

Extract Entity Names

Purpose: Identify financial entities (e.g., companies, individuals) involved in each transaction.

## How It Works:

* + Use a pre-trained BERT model (**XLM-RoBERTa**) for Named Entity Recognition (**NER**) to extract entities from both structured and unstructured text. BERT, a transformer-based model, excels at understanding context, making it ideal for identifying entity names in varied formats.
  + Fine-tune BERT on a financial dataset (**In future**) (e.g., using Hugging Face’s transformers library) to improve accuracy for transaction-specific entities.

Implementation:

* + Preprocess the text (tokenize, remove noise) using transformers.
  + Pass each transaction to the BERT NER model (**XLM-RoBERTa**) to tag entities like "ABC Corp" as organizations.
  + Output a list of entities per transaction (e.g., {txn1: ["ABC Corp"], txn2: ["XYZ Ltd"]}).

Tools:

* Hugging Face Transformers ([Hugging Face Transformers](https://huggingface.co/)), another alternative we explored is spaCy ([spaCy](https://spacy.io/" \t "_blank)).

OFAC Check

Purpose:

Screen entities against the U.S. Office of Foreign Assets Control (OFAC) sanctions list to identify compliance risks.

## How It Works:

* + Use an OFAC API (e.g., Treasury’s SDN list API. (Another alternative we explored is third-party service like Refinitiv) to check if extracted entities are flagged.
  + If an entity appears on the list, assign a "High" risk score immediately due to regulatory implications, bypassing further analysis for that entity.

## Implementation:

* + Send a GET or POST request to the OFAC API with the entity name (e.g., requests.get("<https://scsanctions.un.org/resources/xml/en/consolidated.xml>")).
  + Parse the response (e.g., XML) to determine if the entity is flagged (e.g., "match": true).
  + Update the risk score: risk\_score = "High" if flagged, else proceed to the next step.
  + High in the step above means 0.90 on risk score.

Tools: Python requests library, OFAC API (e.g., [Treasury OFAC API](https://home.treasury.gov/policy-issues/financial-sanctions/specially-designated-nationals-list-sdn-list)).

News and Web Search

Purpose**:** Assess reputation risk by analyzing recent news about the entities.

## How It Works:

* + Query the NewsAPI for articles mentioning the entity (e.g., "ABC Corp") over a recent period (e.g., last 30 days).
  + Apply a BERT-based sentiment classifier to determine if the news reflects negative, neutral, or positive sentiment, which informs the risk score.

## Implementation:

* + Fetch news using NewsAPI (requests.get("[https://news.google.com/rss/search?q={entity\_name}](https://news.google.com/rss/search?q=%7bentity_name%7d)")).
  + Preprocess article text and classify sentiment using a BERT model (e.g., bert-base-uncased fine-tuned for sentiment analysis).
  + Assign a preliminary risk score: e.g., "High" for negative sentiment, "Medium" for neutral, "Low" for positive.

Tools: GoogleNews API or we can also use NewsAPI ([NewsAPI](https://newsapi.org/" \t "_blank)), Hugging Face Transformers.

# SEC EDGAR for Anomalies Using a Classifier Model

## Purpose:

Identify financial or compliance risks from SEC filings for public companies.

## How It Works:

* + Access SEC EDGAR filings (e.g., 10-K, 10-Q reports) for the entity using an API.
  + Use a classifier (e.g., BERT or a simpler ML model like Logistic Regression) to detect anomalies or risk indicators, such as mentions of litigation, debt issues, or regulatory scrutiny.

## Implementation:

* + Fetch filings via the EDGAR API "<https://efts.sec.gov/LATEST/search-index>" and [https://data.sec.gov/submissions/CIK{cik}.json](https://data.sec.gov/submissions/CIK%7Bcik%7D.json)" .
  + We can also scrape from [SEC EDGAR](https://www.sec.gov/edgar) using requests and BeautifulSoup.
  + Preprocess text and classify sections for risk signals (e.g., "lawsuit" → high risk, "stable earnings" → low risk).
  + Output risk observations (e.g., {entity: "ABC Corp", anomalies: ["litigation pending"]}).

Tools: SEC EDGAR API, BeautifulSoup ([BeautifulSoup](https://www.crummy.com/software/BeautifulSoup" \t "_blank)), scikit-learn ([scikit-learn](https://scikit-learn.org/)).

# Create Augmented Input for LLM

Purpose: Combine data from NewsAPI and SEC EDGAR to provide the LLM with comprehensive context for risk assessment.

## How It Works:

* + Aggregate the transaction data, news sentiment scores, and EDGAR anomaly observations into a structured input (e.g., JSON or plain text).
  + Format the input to include all relevant details for each entity, ensuring the LLM can process it effectively.

## Implementation:

* + We pass extracted data from News api and Sec Adgar so that our LLM can give us real time analysis on these data.
  + Example data for LLM:

*data = {*

*"Transaction ID": "TXN-2023-5ABD",*

*"Extracted Entity": ["Wells Fargo", "Global FinTech Corp"],*

*"Entity Type": ["Corporation", "Corporation"],*

*"News API": [*

*{"Entity": "Wells Fargo", "Risk Score": 0.4, "Confidence Score": 0.6, "Reason": "https://news.example.com/wells-fargo"},*

*{"Entity": "Global FinTech Corp", "Risk Score": 0.3, "Confidence Score": 0.7, "Reason": "https://news.example.com/global-fintech"}*

*],*

*"Edgar API": [*

*{"Entity": "Wells Fargo", "Risk Score": 0.35, "Confidence Score": 0.65, "Reason": "SEC Filing XYZ"},*

*{"Entity": "Global FinTech Corp", "Risk Score": 0.25, "Confidence Score": 0.75, "Reason": "SEC Filing ABC"}*

*]*

*}*

* + LLM receives this prompt and provide us output in desired format

# Push Data to LLM for a Refined Risk Assessment

Purpose: Use an LLM to synthesize the augmented input and generate a detailed, context-aware risk score in a specific format.

## How It Works:

* + Send the augmented input to an LLM (using Groq api which uses LLAMA 3 8b parameter) with a structured prompt. We also explored on local Gemma 3.0 trained with unsloth. Found hosted Grok api is highly responsive but we can do it locally as well if we have good GPU.
  + The LLM interprets the data, weighs the risk factors (e.g., litigation > news sentiment), and outputs a risk score and explanation in a predefined format.

## Implementation:

* + Prompt example:
  + Below is the example prompt for LLM using above data and desired output format

*PROMPT\_TEMPLATE = """*

*You are an AI model designed for financial risk assessment. Given a transaction containing extracted entities*

*and their associated risk scores from multiple sources (News API, SEC EDGAR), along with your base trained knowledge,*

*your task is to generate a structured JSON response evaluating the risk.*

*Instructions:*

*- Maintain the same Transaction ID, Extracted Entity, and Entity Type from the input.*

*- Combine multiple risk scores from News API, SEC EDGAR, and your base trained knowledge using a weighted average approach to evaluate the final risk score.*

*- Extract supporting evidence for the risk score from news reports, filings, and your trained base intelligence.*

*- Clearly justify the reason for risk classification, including any specific red flags, past regulatory actions, or controversies.*

*- Include citation links from the News API's reason key in the final JSON response.*

*- Using the combined knowledge update the 'Supporting Evidence' field combining all the information from all the sources.*

*- \*Respond strictly in a valid JSON format with no extra text or explanations. Do not include markdown formatting (e.g., ```json).\**

*- In the response, dont include "Here is the generated JSON response:", or any other description apart from the json*

*Input JSON:*

*{input\_data}*

*Expected JSON Output Response Format:*

*{{*

*"Transaction ID": "{transaction\_id}",*

*"Extracted Entity": {extracted\_entity},*

*"Entity Type": {entity\_type},*

*"Risk Score": ,*

*"Confidence Score": ,*

*"Supporting Evidence": ["Google news", "Edgar", "LLM Knowledge"],*

*"Reason": {{*

*"entity1": "Explanation with risk score - source link",*

*"entity2": "Explanation with risk score - source link"*

*}}*

*}}*

*Now, generate the JSON response strictly following the format above.*

*"""*

## Tools:

Groq api which uses LLAMA 3 8b, Unsloth with Gemma 3 1b.

# Return the Result to the UI via API

## Purpose:

Deliver the final risk assessment to the user through the Streamlit UI.

## How It Works:

* + FastAPI receives the LLM’s output, formats it as a JSON response, and sends it back to the Streamlit frontend.
  + Streamlit displays the result in a user-friendly format, such as a table or card showing entity names, risk scores, and reasons.

## Implementation:

* + FastAPI endpoint (e.g., /upload).
  + Response:

Non Risk Response:

*{*

*"Transaction ID": "TXN-2021-7653",*

*"Extracted Entity": [*

*"ABCD",*

*"Wells Fargo",*

*"Inc."*

*],*

*"Entity Type": [*

*"Corporation",*

*"Corporation",*

*"Corporation"*

*],*

*"Risk Score": 0.38,*

*"Confidence Score": 0.85,*

*"Supporting Evidence": [*

*"Google news",*

*"Edgar",*

*"LLM Knowledge"*

*],*

*"Reason": {*

*"ABCD": "Wells Fargo: financial fraud (0.43) - https://news.google.com/rss/articles/CBMilwFBVV95cUxNTXBmWUQxM1Bva2FrM2xFVi1mQXFiMzZnNElrQWY1a0xpTV83RW4tZjRScGlpbWluSEtUcUxaSW90RDdIMW1zelRMeU5LLWp2ekU5SnhGOUp0VmdVZjdFQ0d4a1g3QmhCd0Rra0FFLVV6d2VsT2t6ZVh1TlZaSTE3ZnJrQUZGLV9pdW90VFUwMzAyMkxBdDdR0gGcAUFVX3lxTFBCT0hSOU81VmtGZnF0UXBkRjd3bHU1enU1Q2hfb1daV3dWSlByRktUbGFYZnhiblNQRVdsdGg0ZzZQM29Kd21CSGQ2TWNheUxWY1g3X1NSX1hUZW1RaFI4ajZrS3Fua3lkMnU2M1VweUNsQm95MHR0ZzE4NnJla3NCbndaNnB1MENLZkc2UUxLN3haVWx3LUV4a0xUQw?oc=5",*

*"Wells Fargo": "No SEC records found for ABCD. | Inc. filed 4 on Unknown Date: Discloses insider trading, which could indicate undisclosed risks or potential misconduct. | Inc. filed 144 on Unknown Date: Indicates insider stock sales, which may suggest lack of confidence in the company. | Inc. filed 4 on Unknown Date: Discloses insider trading, which could indicate undisclosed risks or potential misconduct. | Inc. filed 144 on Unknown Date: Indicates insider stock sales, which may suggest lack of confidence in the company. | Inc. filed 4 on Unknown Date: Discloses insider trading, which could indicate undisclosed risks or potential misconduct."*

*}*

*}*

* + Risk Response:

*[*

*{*

*"Transaction ID": "TXN-2023-5ABDSender",*

*"Extracted Entity": [*

*"Horizons consulting LLC",*

*"Swiss Bank",*

*"BOSCO GILAN",*

*"Cayman National Bank",*

*"Ali Al",*

*"Mansoori",*

*"MUNITIONS INDUSTRY DEPARTMENT",*

*"Quantum Holding Ltd",*

*"Global FinTech Corp",*

*"Richard Branson",*

*"Amazon Inc"*

*],*

*"Entity Type": [*

*"Corporation",*

*"Corporation",*

*"Corporation",*

*"Corporation",*

*"Individual",*

*"Individual",*

*"Corporation",*

*"Corporation",*

*"Corporation",*

*"Individual",*

*"Corporation"*

*],*

*"Risk Score": 0.9079,*

*"Supporting Evidence": [*

*"OFAC US Sanctions"*

*],*

*"Confidence Score": 0.95,*

*"Reason": "The following entities have been identified in the OFAC US sanctions list: ["BOSCO GILAN", "MUNITIONS INDUSTRY DEPARTMENT"]. Engaging with sanctioned entities poses regulatory, financial, and reputational risks."*

*},*

*{*

*"Transaction ID": "TXN-2025-1831Sender",*

*"Extracted Entity": [*

*"ABDUL KABIR MOHAMMAD",*

*"JAN",*

*"Swiss Bank",*

*"Dark Future Nonprofit Inc",*

*"Cayman National Bank",*

*"Ali Al",*

*"Mansoori",*

*"Sarah Johnson",*

*"Quantum Holding Ltd",*

*"Global FinTech Corp",*

*"Richard Branson",*

*"Amazon Inc"*

*],*

*"Entity Type": [*

*"Individual",*

*"Individual",*

*"Corporation",*

*"Corporation",*

*"Corporation",*

*"Individual",*

*"Individual",*

*"Individual",*

*"Corporation",*

*"Corporation",*

*"Individual",*

*"Corporation"*

*],*

*"Risk Score": 0.8,*

*"Supporting Evidence": [*

*"OFAC UN Sanctions"*

*],*

*"Confidence Score": 0.95,*

*"Reason": "The following entities have been identified in the OFAC UN sanctions list: ["ABDUL KABIR MOHAMMAD", "JAN"]. Engaging with sanctioned entities poses regulatory, financial, and reputational risks."*

*},*

*{*

*"Transaction ID": "TXN-2023-5ABD",*

*"Extracted Entity": [*

*"Wells Fargo",*

*"Global FinTech Corp"*

*],*

*"Risk Score": 0.7044098973274231,*

*"Supporting Evidence": [*

*"Google News"*

*],*

*"Confidence Score": 0.7044098973274231,*

*"Reason": "Wells Fargo: financial fraud (0.70) - https://news.google.com/rss/articles/CBMicEFVX3lxTE1XTGZGakpaNFVZWDRPd1RIcE9ZTzdYcmsyWFZtR0dOUzJMY2NKTWh0UmtBUXpiNjhHOGNsYm5PYThSUjNiS2lJa3hrMnB3eGU4eDd0Snl4Zk16eUNaSjhkazNZbmI5UTN0U3hkTzVnSDE?oc=5"*

*}*

*]*

Streamlit uses st.write() or st.table() to render the results. We can also use PDF or

## Tools:

FastAPI, Streamlit.

# Additional Considerations

* **Data Privacy**: We need to ensure compliance with GDPR ([European Commission GDPR](https://ec.europa.eu/info/law/law-topic/data-protection_en)) or CCPA when handling transaction data.
* **Scalability**: We deployed the system using Google Colab, leveraging ngrok to expose both FastAPI and Streamlit services. First, we installed the required dependencies and created a FastAPI backend, which was run in a separate thread. We then used ngrok to generate a public URL for FastAPI. Similarly, we developed a Streamlit frontend and exposed it using ngrok. This setup allowed us to test and interact with the application without requiring a dedicated cloud environment.
* **Bias:** Validate LLM outputs against ground truth to avoid skewed risk scores from biased training data.

Tools and Resources Summary

* UI and API: Streamlit, FastAPI
* NLP: BERT (Hugging Face Transformers)
* APIs: OFAC API, NewsAPI, SEC EDGAR API
* ML: scikit-learn for classifiers
* LLM: OLLAMA 3 8b, Gemma 3 1b
* *Web Scraping: BeautifulSoup, requests (not used)*

Conclusion

This workflow creates a robust risk analysis system by integrating transaction data ingestion, entity extraction with BERT, compliance checks via OFAC, reputation analysis with NewsAPI, anomaly detection with SEC EDGAR, and final synthesis with an LLM. Streamlit and FastAPI ensure a seamless user experience, while the hybrid approach of NLP, ML, and GenAI balances qualitative and quantitative risk assessment. This system is adaptable for financial entities of any type, providing actionable risk scores based on transaction data.

Key Citations

* [Streamlit Docs](https://docs.streamlit.io/)
* [FastAPI Docs](https://fastapi.tiangolo.com/)
* [Hugging Face Transformers](https://huggingface.co/)
* [Treasury OFAC API](https://home.treasury.gov/policy-issues/financial-sanctions/specially-designated-nationals-list-sdn-list)
* [NewsAPI](https://newsapi.org/)
* [SEC EDGAR](https://www.sec.gov/edgar)
* [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup)
* [scikit-learn](https://scikit-learn.org/)
* [OpenAI API](https://platform.openai.com/docs)
* [LangChain](https://github.com/langchain-ai/langchain)
* [European Commission GDPR](https://ec.europa.eu/info/law/law-topic/data-protection_en)