**Multi-Language Piracy Text Detection**

This application identifies piracy-related keywords in texts or PDFs and translates non-English text to English for efficient keyword detection. It provides a modern interface and stores past prompt history locally.

**Features**

1. Detects piracy keywords in user-provided text or uploaded PDFs.
2. Identifies the language of the input text.
3. Translates non-English text to English using Google Translator API.
4. Stores search history locally for future reference.
5. Supports modern, responsive UI for better user experience.

**Approach**

**Backend**

1. **API Design**:
   * Created a backend API using FastAPI to handle inputs in two formats: text and PDF files.
2. **Text Extraction**:
   * For PDF inputs, implemented a text extraction mechanism using the PyPDF2 library.
3. **Language Translation**:
   * Used the Google Translate API to convert the extracted or provided text into English.
4. **Text Processing**:
   * Applied Natural Language Processing (NLP) techniques to tokenize the text into individual words or phrases.
5. **Keyword Matching**:
   * Compared each token against a pre-defined dictionary of piracy-related keywords.
   * Marked matches to identify potential piracy indicators.
6. **Response Generation**:
   * Returned a structured response with the matched keywords and their frequency or locations in the text.

**Frontend**

1. **Form Creation**:
   * Built a simple web interface with a form that included:
     + A text input box for plain text.
     + A file upload option for PDFs.
     + A submit button to send the data to the backend.
2. **Integration**:
   * Integrated the frontend with the backend API to allow users to submit data and display results dynamically.
3. **Results Display**:
   * Processed the API response to show a user-friendly report of the detected keywords and their insights (e.g., frequency, context).

**Step 1: Basic Frontend and API Integration**

* **Implementation**: A simple input prompt with a submit button sends text to a backend API that detects piracy keywords.
* **Challenges**:
  + **Language Detection**: The system lacked the ability to identify the language of the input text accurately.
  + **UI Design**: The initial interface was basic and lacked user-friendly features.
  + **History Management**: No functionality for storing or displaying past searches.

**Step 2: Enhanced UI and Additional Features**

* **Improvements**:
  + Redesigned the frontend to display:
    - Detected language.
    - Original and translated text.
    - Keywords indicating piracy.
    - Search history stored in local storage.
* **Challenges**:
  + **Translation Limitations**:
    - Using **Helsinki-NLP/opus-mt-mul-en** for translation was ineffective for short texts (e.g., Spanish phrase "transmitiendo sin permiso").
    - Some languages like Ossetic and Arabic were unsupported.
  + **Examples**:
    - Ossetic: æнæаргъæй райс кинонывтæ торренттæй
    - Arabic: قم بتنزيل أحدث الأفلام مجانًا باستخدام التورنت أو الروابط المغناطيسية.
* **Solution**:
  + Integrated Google Translation API for reliable and extensive language support.

**Step 3: File Upload and PDF Processing**

* **Implementation**: Added a feature to upload PDF files for piracy keyword detection.
* **Challenges**:
  + Extracting text from PDFs on the frontend using libraries like pdfjs-dist failed for many documents.
  + Backend API attempts to extract PDF text also faced inconsistencies.
* **Solution**:
  + Used PyPDF2 in the Flask backend to extract text directly from uploaded PDFs and processed it for piracy keyword detection.

**Backend API doc.**

**Endpoint: /process**

* **With Text Input:** Send a POST request with JSON body { "text": "Your input text" } to detect piracy-related keywords and translate non-English text if necessary.
* **With PDF File:** Send a POST request with a PDF file as form-data (key: file) to extract text, detect keywords, and translate if needed.

**Resources Used**

* **Backend**:
  + Flask: RESTful API framework.
  + PyPDF2: Extract text from PDFs.
  + spaCy: Advanced natural language processing.
  + langdetect: Detect language of the input text.
  + Google Translator API (via RapidAPI): For language detection and translation.
  + Pycountry: Fetch language names from codes.
  + Python3
* **Frontend**:
  + React: UI development.
  + Axios
  + Vite
* **APIs**
  + [Google Translator API on RapidAPI](https://rapidapi.com/IRCTCAPI/api/google-translator9)