**EEG**

### **PDF Data Extraction System**

### **Overview**

This project is a comprehensive data extraction and processing system built using FastAPI for handling PDF files related to EEG data analysis. The system processes PDF files, extracts specific sections of data, calculates key metrics, and consolidates the results into CSV and Word documents for further analysis.

### **Components**

The system is divided into three main modules:

1. **api.py** - The primary FastAPI application that provides endpoints for file uploads, data processing, and calculation requests.
2. **calc.py** - A module responsible for calculating metrics and generating reports based on CSV data extracted from PDFs.
3. **pdf\_extraction.py** - The core extraction module that reads PDF files, identifies relevant sections, and extracts specific data points for processing.

### **Detailed Analysis**

#### **1. api.py**

* **Objective:** To provide endpoints for uploading PDFs, processing data, and generating consolidated reports.
* \*\*Endpoints:"  
  + /extraction (POST): Accepts two PDF files, extracts data using the pdf\_extraction.py module, calculates relevant metrics, and generates a CSV file and a DOCX report.
  + /calculation (POST): Accepts a CSV file and a section type, calculates band-wise absolute sums using the calc.py module, and returns the results.
* **Core Functions:**
  + extract\_data: Handles the extraction of data from two PDFs, processes them using extract\_coherence\_phase\_lag and extract\_other\_sections from pdf\_extraction.py, and consolidates the data into CSV and DOCX files.
  + create\_combined\_document: Combines the data from multiple CSV files into a single Word document, structuring the data by subsection and band.
  + bandwise\_absolute\_sum: Calculates absolute sums for each band in a given CSV file, categorized by subsections and bands.
* **Complexities:**
  + Handling multiple file uploads and maintaining a structured temporary directory.
  + Integrating data extraction, processing, and report generation into a single endpoint.
  + Ensuring data integrity and preventing data loss in case of unexpected errors.

#### **2. calc.py**

* **Objective:** To handle all metric calculations and document generation based on the extracted data.
* **Functionalities:**
  + compute\_subsection\_bandwise\_metrics: Computes metrics such as absolute sums, average values, delta, and percent change for each band and subsection.
  + save\_to\_docx: Saves the computed metrics into a DOCX document with a structured layout.
  + load\_csv: Utility function for loading CSV files for processing.
* **Complexities:**
  + Handling potential data inconsistencies in the CSV files (e.g., missing values, non-numeric data).
  + Ensuring accurate calculation of delta and percentage change while managing zero-division errors.

#### **3. pdf\_extraction.py**

* **Objective:** To extract data from PDFs based on predefined sections and bands.
* **Data Sections Processed:**
  + Z Scored FFT Absolute Power
  + Z Scored FFT Power Ratio
  + Z Scored Peak Frequency
  + Z Scored FFT Coherence
  + Z Scored FFT Phase Lag
* **Data Organization:**
  + Data is categorized into LEFT, RIGHT, and HOMOLOGOUS subsections, allowing for specific intrahemispheric and interhemispheric analysis.
  + Band mappings are defined for each section to maintain consistency in the extraction process.
* **Core Functions:**
  + extract\_coherence\_phase\_lag: Extracts coherence and phase lag data from PDFs, organizing it by LEFT, RIGHT, and HOMOLOGOUS subsections.
  + extract\_other\_sections: Extracts Absolute Power, Power Ratio, and Peak Frequency data, segregating it by LEFT, RIGHT, and CENTER sections.
  + merge\_data: Consolidates extracted data from multiple PDFs, ensuring consistent data structure for downstream processing.
  + save\_to\_csv: Saves extracted data to CSV files, categorized by section and subsection.
* **Complexities:**
  + Handling complex PDF structures with varying formats and section headers.
  + Managing missing or malformed data during extraction.
  + Optimizing data processing to handle large files efficiently.