README

Overview

This document outlines the implementation of a text extraction and web scraping program that pre-processes the data extracted from scanned documents and stores them in a MySQL database.

1. **extract.py**
    - The provided Python script utilizes PyPDF2 and pdf2image libraries to extract text from PDF files.
    - It defines functions to extract text from PDFs, parse the extracted text using regular expressions, and perform OCR on PDFs using pytesseract.
    - The script aims to handle PDF text extraction, parsing, and OCR to extract relevant information from the files.

2. **webscraper.py**
    - The script utilizes the Selenium library to automate web interactions, specifically to retrieve well information from the "https://www.drillingedge.com" website.
    - It connects to a MySQL database named "raydiant" using the mysql.connector library, adds new columns to the 'data' table, and retrieves API numbers and well names from the table for further processing.
    - The functions search_and_insert_well_info and insert_into_data_table are defined to search for well information on the website based on API numbers and well names and then insert the retrieved data into the 'data' table.
    - The script exports the entire 'data' table to a CSV file named "data_table.csv" after successfully retrieving and processing the well information, providing a convenient way to store and analyze the collected data.

3. **upload.py**
    - The code establishes a connection to a MySQL database using the mysql.connector library, including functions to connect, create a table, and insert data.
    - It defines a table named 'data' with various columns and data types, checks if the table already exists, and creates it if not. The table is designed to store information related to well data.
    - The code reads well data from a CSV file named 'well_info.csv' using pandas, connects to MySQL, ensures the 'data' table is created, and inserts the well data into the MySQL database. The script handles date format conversion and NaN values appropriately during data insertion.

4. **main.py**
    - The script processes PDF files in a specified folder, extracting text, performing regular expression-based parsing using predefined patterns, and saving the extracted data to a CSV file named "well_info.csv."

- The script uses functions from external modules, namely extract.py, upload.py, and webscraper.py. The functionalities include text extraction from PDFs, uploading data to a MySQL database, and web scraping.
- The regex_patterns dictionary contains regular expressions for extracting specific data fields from the text extracted from PDF files. These patterns are used in the parse_extracted_text function to identify and extract relevant information.
- The script utilizes a loop to iterate through PDF files in the specified input folder, extract text, apply regular expression parsing, and save the results to a CSV file. Additionally, it checks for the existence of the CSV file and, if present, skips further processing, indicating completion of the extraction process. The loop continues until the CSV file is created, suggesting completion of text extraction and parsing for all PDF files in the input folder.

Prerequisites

- MySQL server installed and running.
- Python 3.x
- Selenium WebDriver (Chrome)
- Required Python libraries: mysql.connector, pytesseract, PyPDF2, pdf2image, selenium, pandas

Usage

- **`Python3 main.py`**
- **`Python3 upload.py`**
- **`Python3 webscrapper.py`**

Output

- Extracted data is stored in MySQL database