Extracting Tables from PDFs and Webpages using Python

Overview

This script extracts tables from **PDF files** or **web pages** and saves them as CSV files. It allows users to:

- Extract tables from a **local PDF file** using `pdfplumber`
- Extract tables from an **HTML page** using `BeautifulSoup`
- **Filter tables** based on a keyword found in column names
- **Select a specific table** by index (0-based)
- Save the extracted **filtered tables** to a specified folder

---

Code Breakdown

1. Main Function: `extract_table()`

This function orchestrates the extraction process based on the input type.

Parameters:

- `input_path_or_url` *(str)*: Either a **local PDF file path** or a **URL** to a webpage.
- `table_index` *(int or None)*: If specified, extracts only the table at this 0-based index after filtering.
- `key_words` *(str or None)*: If provided, only tables containing this keyword in the column names are returned.
- `output_folder` *(str, default="output_tables")*: Folder where extracted tables will be saved as CSV files.

Process:

1. **Check if input is a file or a URL**
   - If it's a **PDF file**, call `_extract_from_pdf()`.
   - If it's a **webpage**, call `_extract_from_web()`.
   - If neither, print an error and return.
2. **Normalise column names**
   - Convert all column names to lowercase and remove newline characters (`\n`) to improve keyword filtering.
3. **Filter by keyword**
   - If `key_words` is set, check if any column contains this keyword.
4. **Select the specific table index**
   - If `table_index` is specified, extract that particular table.
5. **Save the filtered table(s) as CSV**

- Call `_save_to_csv()` to store the extracted table(s).

---

2. Extracting Tables from a PDF: `_extract_from_pdf()`

- Opens the PDF using `pdfplumber`.
- Iterates through each page and extracts tables.
- Converts extracted tables into Pandas DataFrames.
- Returns a list of DataFrames.

---

3. Extracting Tables from a Webpage: `_extract_from_web()`

- Uses `requests` to download the webpage content.
- Parses the HTML using `BeautifulSoup`.
- Searches for `<table>` elements, extracts rows, and stores them in DataFrames.
- Returns a list of DataFrames.

---

4. Saving Tables as CSV: `_save_to_csv()`

- Creates the output folder if it doesn't exist.
- Saves each extracted table as a CSV file with names like `filtered_table_1.csv`, `filtered_table_2.csv`, etc.
- Prints confirmation messages for saved files.

Example Usage

Extracting a Table from a PDF:

```
pdf_tables = extract_table("sample.pdf", key_words="Purpose of con
tribution", table_index=1)
```

In [ ]:

```
import os
import pdfplumber
import pandas as pd
import requests
from bs4 import BeautifulSoup

def extract_table(input_path_or_url, table_index=None, key_words=None, outp
    """
    Extracts tables from a given PDF or webpage and saves them as CSV files

    Parameters:
    input_path_or_url (str): Path to the PDF file or URL.
    table_index (int or None): Table index to extract (0-based). If None, r
    key_words (str or None): If set, extracts only tables containing the ke
    output_folder (str): Folder name to save the extracted table(s).

    Returns:
    list of pandas.DataFrame: Extracted tables as DataFrames.
    """
    extracted_tables = []

    if os.path.isfile(input_path_or_url):  # Input is a file
        file_ext = os.path.splitext(input_path_or_url)[-1].lower()

        if file_ext == ".pdf":
            extracted_tables = _extract_from_pdf(input_path_or_url)
        else:
            print("Unsupported file format. Only PDFs are supported.")
            return []

    elif input_path_or_url.startswith("http"):  # Input is a URL
        extracted_tables = _extract_from_web(input_path_or_url)

    else:
        print("Invalid input. Provide a valid file path or URL.")
        return []

    if not extracted_tables:
        print("No tables extracted from the document.")
        return []

    # Normalise column names and print extracted columns for debugging
    for i, df in enumerate(extracted_tables):
        df.columns = [col.strip().replace("\n", " ").lower() if isinstance(
        print(f"Table {i+1} Columns: {df.columns.tolist()}")

    # Filter by keyword if provided
    if key_words:
        key_words_lower = key_words.lower()
        extracted_tables = [df for df in extracted_tables if
                            any(key_words_lower in col for col in df.column

    if not extracted_tables:
        print(f"No tables matched the keyword: {key_words}")
        return []

    # Apply table index filtering
    if table_index is not None and len(extracted_tables) > table_index:
        extracted_tables = [extracted_tables[table_index]]

    # Save only filtered tables
```

```
    _save_to_csv(extracted_tables, output_folder)
    return extracted_tables

def _extract_from_pdf(pdf_path):
    """Extracts tables from a PDF file."""
    tables = []
    with pdfplumber.open(pdf_path) as pdf:
        for page_num, page in enumerate(pdf.pages):
            extracted = page.extract_table()
            if extracted:
                df = pd.DataFrame(extracted[1:], columns=extracted[0])
                tables.append(df)
    return tables

def _extract_from_web(url):
    """Extracts tables from a webpage."""
    try:
        response = requests.get(url)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, 'html.parser')
        tables = []

        for table in soup.find_all('table'):
            rows = table.find_all('tr')
            data = [[cell.get_text(strip=True) for cell in row.find_all(['t
            if data:
                df = pd.DataFrame(data[1:], columns=data[0])
                tables.append(df)

        return tables

    except Exception as e:
        print(f"Error fetching webpage: {e}")
        return []

def _save_to_csv(tables, output_folder):
    """Saves extracted tables to CSV files."""
    os.makedirs(output_folder, exist_ok=True)
    for idx, df in enumerate(tables):
        csv_filename = os.path.join(output_folder, f"filtered_table_{idx+1}
        df.to_csv(csv_filename, index=False)
        print(f"Table saved to {csv_filename}")

# Input
input = ""


# Extract tables from PDF
tables = extract_table(input, key_words="", table_index=None)
```