Python
PROGRAMMING

CONVERTING PDF TO TEXT

Agentiver
Academy

Tutorial: Creating a PDF to Text Converter with Python

Step 1: Setting Up Your Environment

Before you begin, ensure you have Python installed on your system. You'll also need the following Python libraries:

- **Tkinter**: for creating the GUI (Graphical User Interface) to select folders.
- **PyPDF2**: for reading and extracting text from PDF files.

If you don't have these libraries installed, you can install PyPDF2 using pip:

pip install PyPDF2

Note: Tkinter comes pre-installed with Python, so you don’t need to install it separately.

Step 2: Importing Necessary Libraries

Start by importing the required libraries:

python

```
import os
import tkinter as tk
from tkinter import filedialog
from PyPDF2 import PdfReader
```

- os: Provides functions for interacting with the operating system.
- tkinter and filedialog: Used to create a simple GUI for folder selection.
- PyPDF2.PdfReader: Used to read PDF files and extract text.

Step 3: Writing Helper Functions

You need a few helper functions to handle the text processing, such as determining if a line belongs to the previous one or if it's a heading.

1. **belongs_to_previous_line(previous_line, current_line)**:
    - This function checks if the current line should be appended to the previous line.
    - It assumes that if the previous line ends with a sentence-ending punctuation (e.g., '.', '!', '?') and the current line starts with a lowercase letter, the lines should be combined.
    -

```
def belongs_to_previous_line(previous_line, current_line):
    if previous_line.endswith(('.','!','?')):
        return False
    if current_line and current_line[0].islower():
        return True
    return False
```

2. **is_title_case(line)**:

- This function checks if a line is in title case (i.e., each word starts with a capital letter).

```
def is_title_case(line):
    return line == line.title()
```

3. **is_heading(line)**:
    - This function considers a line as a heading if it has fewer than five words and is in title case.

```
def is_heading(line):
    return len(line.split()) < 5 and is_title_case(line)
```

Step 4: Writing the Main Conversion Function

The main function, convert_pdf_to_text, handles reading the PDF, processing each line, and saving the output as a text file.

python

```
def convert_pdf_to_text(pdf_file, txt_file):
    with open(txt_file, 'w', encoding='utf-8') as text_file:
        reader = PdfReader(pdf_file)
        text = ""
        for page in reader.pages:
            lines = page.extract_text().splitlines()
            paragraph = ""
            for i, line in enumerate(lines):
                line = line.strip()
                if is_heading(line):
                    if paragraph:
                        text += paragraph + "\n"  # Add the previous paragraph
                    text += line + "\n"  # Treat as a heading, add on its own line
                    paragraph = ""  # Reset paragraph
                else:
                    if i > 0 and belongs_to_previous_line(paragraph, line):
                        paragraph += " " + line  # Concatenate to previous line
                    else:
                        if paragraph:
                            text += paragraph + "\n"
                        paragraph = line
            if paragraph:  # Add the last paragraph from the page
```

```
            text += paragraph + "\n"
        text_file.write(text)
    print(f"Conversion complete. The text file is saved as {txt_file}")
```

Step 5: Creating the GUI for Folder Selection

Next, you need to create a GUI for selecting the folders. Tkinter is perfect for this because it's simple and easy to use.

```
root = tk.Tk()
root.withdraw()

pdf_folder_path = filedialog.askdirectory(title="Select Folder with PDF Files")
save_folder_path = filedialog.askdirectory(title="Select Folder to Save Text Files")
```

- root.withdraw() hides the main Tkinter window since we only need the file dialog.
- filedialog.askdirectory() opens a dialog to select a directory.

Step 6: Iterating Through the PDF Files and Converting Them

Finally, loop through all the PDF files in the selected folder, converting each one to a text file and saving it in the specified folder.

python

```
if pdf_folder_path and save_folder_path:
    for filename in os.listdir(pdf_folder_path):
        if filename.endswith(".pdf"):
            pdf_file = os.path.join(pdf_folder_path, filename)
            txt_file = os.path.join(save_folder_path, os.path.splitext(filename)[0] + ".txt")
            convert_pdf_to_text(pdf_file, txt_file)
else:
    print("Folder selection was cancelled.")
```

- This code checks if the user has selected both folders. If they have, it processes each PDF file in the folder.
- The text files are saved in the chosen folder with the same name as the original PDF but with a .txt extension.

Step 7: Running the Script

To run the script:

1. Save the code in a Python file, for example, pdf_to_text_converter.py.
2. Run the script using a Python interpreter.

python pdf_to_text_converter.py

After running the script:

- You will be prompted to select the folder containing the PDF files.
- Then, you'll select the folder where the text files will be saved.
- The script will process each PDF in the selected folder and save the converted text files in the specified location.

Conclusion

You've now created a Python script that efficiently converts PDF files to text files, preserving the structure and headings as much as possible. This tutorial introduced you to essential concepts like text processing and GUI creation with Tkinter.