ChatGPT

Tools and Libraries

Several Python libraries can extract text from a text-based PDF and help parse dictionary entries. For example, **PyMuPDF** (the `fitz` library) can open a PDF and extract all text per page. The PyMuPDF docs show this workflow (install via `pip install PyMuPDF`) [1]:

```
import fitz  # PyMuPDF
doc = fitz.open("dictionary.pdf")
text = ""
for page in doc:
    text += page.get_text()
```

This iterates over each page and concatenates the plain text [2] [3]. Similarly, **pdfplumber** (built on pdfminer.six) can be installed with `pip install pdfplumber` (it automatically pulls in dependencies like Pillow and pdf2image [4]). A typical usage is:

```
import pdfplumber
with pdfplumber.open("dictionary.pdf") as pdf:
    for page in pdf.pages:
        text = page.extract_text()
        # process text...
```

As the pdfplumber docs illustrate, calling `page.extract_text()` returns the text block for that page [5] [6]. The **pdfminer.six** library (install via `pip install pdfminer.six`) can also pull all text from a PDF. For instance, one can use its high-level API:

```
from pdfminer.high_level import extract_text
text = extract_text("dictionary.pdf")
```

as shown in examples [7]. (Pdfminer is the engine behind pdfplumber and offers more control over layout analysis if needed.)

Another approach is to use the pdf_to_json tool, which uses Poppler to convert a PDF into a detailed JSON-like Python dict. After installing (`pip install --upgrade git+https://github.com/antoinecarme/pdf_to_json.git` [8]), you can do:

```
import pdf_to_json as p2j
converter = p2j.pdf_to_json.pdf_to_json_converter()
data = converter.convert("dictionary.pdf")
```

This yields a Python dict containing all pages, text blocks, and attributes [9]. One can then traverse this dict to extract the dictionary entries.

(Other tools like Tabula or Camelot can extract tables if the PDF is laid out in strict columns, but standard bilingual dictionaries usually use inline text per entry rather than formal tables.)

Parsing Entries and Outputting JSON

Once the raw text is extracted, the next step is to parse it into structured fields (word, part of speech, meanings). Typically each dictionary line looks like:

```
word (POS) meaning1; meaning2; ...
```

One can loop over the extracted text lines and use string methods or regex to split out the fields. For example:

```
import re
entries = []
for line in text.splitlines():
    match = re.match(r"^(.+?)\s*\(([^)]+)\)\s*(.+)$", line)
    if match:
        word, pos, defs = match.groups()
        meanings = [m.strip() for m in defs.split(";")]
        entries.append({"word": word, "pos": pos, "meanings": meanings})
```

This builds a list of Python dicts, e.g. `{"word": "apple", "pos": "n.", "meanings": ["a fruit", "a tech company"]}`. Finally, use Python's built-in `json` module to write this list as JSON:

```
import json
with open("dictionary.json","w", encoding="utf-8") as f:
    json.dump(entries, f, ensure_ascii=False, indent=2)
```

Citing common usage, one serializes a dict to JSON and writes it out [10].

Example Input/Output

Example PDF text (one entry):

```
apple (n.) a fruit; a tech company
```

Parsed JSON output:

```
[
  {
    "word": "apple",
    "pos": "n.",
    "meanings": ["a fruit", "a tech company"]
  }
]
```

This shows the desired structure. In practice you'd loop through all extracted lines, skip headers/footers, and handle multi-line definitions if any.

Dependencies and Preprocessing

- **Install libraries:** e.g. `pip install PyMuPDF pdfplumber pdfminer.six`. Pdfplumber will also install **Pillow** and **pdf2image** as needed [4]. The `pdf_to_json` library (if used) is installed via its GitHub URL [8].
- **Preprocessing:** Remove non-entry text (page numbers, headings). If the dictionary uses multi-column pages, one may need to split by x-coordinate (both PyMuPDF and pdfplumber allow inspecting text box coordinates) or use a table parser.
- **Character Encoding:** The extracted text is Unicode (UTF-8). Ensure correct encoding when writing JSON.
- **POS Abbreviations:** Often POS tags (n., v., adj., etc.) are within parentheses – our regex above assumes that format. Adjust as needed for your dictionary's style.

By combining one of these PDF-text libraries with Python text parsing and the `json` module, you can automate converting a standard bilingual dictionary PDF into a structured JSON list of `{word, pos, meanings}` entries.

Sources: Official docs and examples from PyMuPDF [2] [3], pdfplumber [5] [6], pdfminer.six [7], and the `pdf_to_json` GitHub [9]; plus general JSON-writing reference [10]. Each snippet above is adapted from those sources.

---

1 2 PyMuPDF · PyPI
https://pypi.org/project/PyMuPDF/

3 The Basics - PyMuPDF 1.26.0 documentation
https://pymupdf.readthedocs.io/en/latest/the-basics.html

4 Python Libraries to Extract Tables From PDF: A Comparison
https://unstract.com/blog/extract-tables-from-pdf-python/

5 6 How To Easily Extract Text From Any PDF With Python | by Vinicius Porfirio Purgato | Analytics Vidhya | Medium
https://medium.com/analytics-vidhya/how-to-easily-extract-text-from-any-pdf-with-python-fc6efd1dedbe

7 Python by Examples: Extract PDF by PDFMiner.six | by MB20261 | Medium
https://medium.com/@mb20261/python-by-examples-extract-pdf-by-pdfminer-six-246cba6f89b3

8 9 GitHub - antoinecarme/pdf_to_json: Python module to Convert a PDF file to a JSON format
https://github.com/antoinecarme/pdf_to_json

10 Reading and Writing JSON to a File in Python | GeeksforGeeks
https://www.geeksforgeeks.org/reading-and-writing-json-to-a-file-in-python/