ChatGPT

英文博士教科書 PDF 轉 Markdown 翻譯與結構化工具比較報告

簡介與問題說明

將英文博士教科書等大型PDF檔轉換為Markdown格式，並進行精準翻譯與內容結構化，是一項極具挑戰的任務。這種轉換不僅要求忠實保留原始文本的結構（如標題、段落、清單、表格、公式等），還需支援後續製作條列式總結、筆記、教學路線圖（roadmap）與部落格文章等應用。傳統的做法往往涉及繁瑣的人工複製貼上和格式整理；而新的AI工具鏈有望自動化這一流程，提高效率與準確度。

目前使用的流程包括：先利用 Marker 套件將PDF轉換為Markdown，接著使用大型語言模型（如ChatGPT或「Gemini Flash 2.5」）進行文本分析與翻譯。然而，這個流程存在一些問題：ChatGPT雖然總結能力強，但在缺乏約束時可能產生**幻覺**（hallucination），而且一次無法處理過大的上下文，需要將大PDF切成小片段（約2000字）逐段處理；Gemini Flash 2.5 則號稱可處理長達10萬字的上下文，但其翻譯結果相對平淡，難以滿足高品質譯文的需求。

為了解決上述問題，我們調研並比較了多款近期熱門的PDF轉Markdown工具和解決方案，包括開源專案與商業服務。我們將重點分析以下工具： **Marker**, **MinerU**, **MarkItDown**, **pdf2md (Zyocum)**，評估它們在轉換精度、結構保留、OCR能力、段落斷句/多語翻譯支持、大檔處理、輸出格式與適用場景等方面的差異和表現。此外，我們也將探討有無更完善的開源或商業方案，可以整合「PDF轉Markdown → 翻譯 → 結構化 → 摘要/路線圖」的自動化工具鏈，並展望未來結合代理型AI（Agentic AI）與多模態理解的系統，用於更廣泛的文件智能處理。

主要工具比較概覽

首先，我們對比上述四款工具在核心功能上的表現，如下表所示：

工具	轉換精度與結構保留	OCR能力	段落斷句/多語翻譯支持	大檔案處理	輸出格式	適用場景與特點
Marker	轉換準確，支援標題、段落、程式碼、公式等結構；表格處理仍在改進 [1] [2]。	內建OCR（支援Tesseract與Surya引擎）可處理掃描頁面 [3] [4]。	無內建翻譯；按頁面/區塊輸出文本，語義連貫（自動去除頁眉頁腳） [5]。	可處理整本書等大型PDF，但需較多資源；可用作內部服務異步處理 [6]。	Markdown（附圖片檔）和JSON [7] [8]。	適合學術論文、技術書籍等高結構文檔，需高品質排版還原；可自架部署，成本低。
MinerU	轉換精度高，全方位保留文件結構（標題層級、段落、清單、表格、圖像、公式等） [9] [10]。	自動檢測掃描PDF並啟用OCR，支援84種語言文字識別 [11]。	無內建翻譯；能智慧斷行去雜訊（去除頁碼、浮水印等保持語義連貫） [9]。	最佳效能需GPU加速（16GB↑顯存），可解析複雜版面但硬體需求高 [12] [13]。	Markdown（提供「多模態MD」或適合NLP的MD）與JSON，並輸出LaTeX公式、HTML表格等 [10] [14]。	適合科研論文、專利說明書等包含大量公式表格的PDF；強調高品質解析，支持跨平台部署。
MarkItDown	轉換速度快，保留基本結構（標題、清單、連結等），但對複雜元素（如表格）僅提取純文字，可能丟失欄位對齊 [15]。	支援影像OCR（需安裝相應套件）與語音轉錄，可處理圖片和多媒體 [16]。	無翻譯功能；輸出著重於LLM易讀性，非100%人類格式忠實（以減少雜訊為主) [17]。	輕量高效，可一次批量轉換多種文件；處理超大文件時可能因記憶體限制略有簡化（偏重文本內容）。	Markdown純文本（不主動輸出圖片，圖片可另行描述或嵌入OCR結果） [15] [16]。	適合LLM預處理各類資料（Office文檔、網頁、圖像等）；用於快速將檔案轉成Markdown供模型分析，強調簡潔穩定。

工具	轉換精度與結構保留	OCR能力	段落斷句/多語翻譯支持	大檔案處理	輸出格式	適用場景與特點
pdf2md (OpenAI GPT-4 Vision)	利用GPT-4V多模態解析，結構還原度高（如能生成對應Markdown表格[18]），文字內容準確且包含格式。	透過GPT視覺模型實現OCR（效果佳，可識別表格和圖像文字），但需OpenAI API服務支援[19][18]。	可修改提示詞實現多語翻譯（默認不翻譯）；逐頁處理，可能缺乏跨頁上下文連貫，需要後續人工合併長段。	理論上可處理任意長度PDF（逐頁API調用），但成本隨頁數線性增長且受API速率限制。	Markdown（直接由GPT產生，內含提取文字、Markdown標記及GPT對圖表的描述）[18]。	適合高複雜度或非常關鍵的文件，需要最高保真度還原時使用；亦可作其他工具失敗時的補救，但費用高、速度受限。

上述比較表提供了各工具在主要維度上的表現概況。下面我們針對這些維度進行更詳盡的分析與評價，並討論每種工具的優缺點和適用場景。

工具性能差異分析

1. 轉換精度與結構保留

Marker： Marker 專注於將PDF高精度地轉為Markdown和JSON，強調**快速且準確**地還原文本和結構[20]。它不僅能識別標題、段落、清單等基本結構，還有專門模組處理程式碼區塊、公式和表格等元素。例如，Marker會將PDF中的公式轉換為對應的LaTeX格式，以便在Markdown中正確顯示[21]。對於圖片和圖表，Marker則會抽取出獨立的圖像檔案，並在Markdown中以連結形式嵌入[22]。總體而言，Marker能產生**排版良好**且可讀性高的Markdown輸出[8]。不過，社群反映Marker目前在表格處理上還有提升空間：例如儘管能識別表格內容，但在碰到**合併儲存格、表頭對齊**等複雜表格時，可能出現欄位錯置或空單元格等問題[1]。開發者也意識到這點，表示正在改進表格解析演算法[23]。

MinerU： MinerU 是由 OpenDataLab 開發的一站式PDF解析工具，它以**高品質還原**著稱[24]。MinerU 能自動保持原文件的層次和結構，例如**標題階層、段落、列表、圖像、表格及其標題、註腳**等都會完整提取[9]。尤其在公式和表格方面，MinerU做了專門優化：公式會自動轉為LaTeX表示，嵌入Markdown中；表格則盡可能轉為對應的HTML表格或Markdown表格格式，確保行列關係清晰[10]。其近期版本也引入了**標題階層分類**功能，能夠更好地辨識文件中標題的等級（如章節 vs. 小節），構建文件目錄結構[25]。在實測中，MinerU對各種複雜版面的適應性較強，例如帶浮水印的文件、多欄布局、多圖片或多表格同頁等情況，MinerU都能正確解析並維持閱讀順序[26]。整體而言，MinerU提供了**全方位的結構保留**，適合對轉換質量要求極高的情境。

MarkItDown： 微軟的MarkItDown偏重**簡潔快速**的轉換。它覆蓋的文件格式很廣（PDF只是其中之一），主要目的是將各類內容轉為Markdown以供LLM或文字分析使用 [17] [27]。MarkItDown會保留文件中的重要結構如標題、粗體/斜體、列表、連結等，使轉出的Markdown在語義上與原文一致 [28]。然而，MarkItDown並不以高保真為目標；它生成的結果**更多是機器友好而非完全對人類排版友好** [29]。例如，有用戶反映MarkItDown在處理PDF表格時，只提取了表格中的純文字內容，**忽略了欄位結構**（未轉成真正的Markdown表格），導致欄目關係遺失 [15]。同樣地，對於嵌入的公式，MarkItDown沒有像Marker或MinerU那樣專門轉為LaTeX標記，因此可能僅保留其文字形式或圖片形式，缺少語義標記。總的來說，MarkItDown**優先穩定和速度**，確保大部分內容被抓取，但細節格式上可能較「裸骨」（barebones） [15]。

pdf2md (GPT-4 Vision)： 這個方案利用OpenAI的多模態GPT-4模型直接讀取PDF每頁的圖像並生成Markdown。由於GPT-4具有極強的自然語言和視覺理解能力，它在還原文件結構和內容上表現出色。例如，對於清晰的表格，GPT-4能理解行列關係並直接以Markdown表格格式輸出 [18]；對於文字段落，則會準確地保持原有段落結構和格式（如標題層級、項目符號）等。實驗結果顯示，GPT-4轉出的表格Markdown不僅正確還原內容，連**表頭和欄位對齊**都處理得當 [18]。可以說，在轉換精度和結構保留方面，依賴GPT的pdf2md能夠**媲美甚至超過**傳統規則/模型驅動的解析工具，尤其是在文件複雜、多元素混雜的情況下。然而，該方法也有潛在風險：一方面GPT屬生成式模型，極少數情況下可能對模糊不清的內容進行**臆測補全（hallucination）**，雖然這種情況在純轉錄任務中很少見但仍需留意；另一方面如果PDF某些部分超出GPT視覺識別能力（例如極複雜的公式符號或特殊語言文字），輸出可能出錯或遺漏。總體而言，pdf2md在精度上表現突出，是**高保真轉換**的一個解決途徑，只是它將轉換過程外包給大型生成模型，其一致性和可控性不像傳統工具那樣可預期。

2. OCR能力與多語種支持

現實中許多PDF包含掃描頁面或圖片文字，因此OCR（光學字符識別）的能力至關重要。對比工具的OCR表現：

- **Marker：** 如果PDF檔本身是文字型（可擷取文本），Marker優先使用內建的PDF解析（基於pypdfium2）抓取文字 [30]；但若遇到文字無法直接擷取（例如掃描影像或編碼混亂），Marker會自動啟用OCR模式 [31]。它內建對接了兩種OCR引擎：一種是`surya`（一個深度學習OCR模型），另一種是`tesseract`（透過OCRmyPDF封裝） [32]。Marker會將需要OCR的頁面轉成影像送入上述引擎，提取文字後再融入整體輸出。由於支援Tesseract，Marker理論上可利用Tesseract的多語種套件進行多語OCR；實際上，作者在討論中提到有使用者用Marker處理波蘭語PDF [33]。雖然該使用者認為Marker輸出資料品質不如Azure的專業服務 [34]（特別在表格等結構上），但這說明Marker在OCR文字識別多語言上是可行的。**總體而言，Marker的OCR屬於輔助性質**——當內嵌文字不足時才啟用，但有基本的多語種處理能力。

- **MinerU：** MinerU 非常強調OCR功能。它能自動檢測PDF是否為掃描件或文字亂碼，若是，會 **自動開啟OCR**流程 [11]。MinerU內置的OCR支持高達**84種語言**的文字檢測與識別 [11]（背後可能結合了如PaddleOCR等多語模型）。這意味著無論PDF包含英文、中文或其他語種，MinerU理論上都能識別並提取。此外，MinerU還使用版面分析模型輔助OCR，例如透過版面偵測先區分文本區域再OCR，提高準確率 [35] [36]。對於混合內容頁（如同一頁有圖片文字也有可擷取文字），MinerU採用了**混合提取策略** [37]：優先使用精確的文字模式擷取內容，同時運行OCR獲取更精細的區域定位，最後結合兩者優點，既保證內容準確又保留版面佈局資訊 [37]。因此在OCR能力上，MinerU是**最全面也最強大**的，適用於多語種和各類掃描品質的PDF。

- **MarkItDown：** MarkItDown本身並非專門的OCR工具，但它提供了處理影像和PDF中圖片的能力。對於輸入為圖片或PDF中的嵌入圖片，MarkItDown能調用OCR來提取文字 [16]。默認情況下，它可能採用Python的OCR庫（如EasyOCR或Tesseract等）來完成這一任務，因此支持的語言取決於所用OCR庫的能力。如果使用者需要更強的影像識別，MarkItDown還提供與LLM結合的方案：可以設定一個`llm_client`與模

型，讓LLM來生成圖片描述 [38]。例如，對於PDF中的圖表，MarkItDown可選擇將圖像發給一個圖像描述模型或GPT，以獲取圖像中的資訊並在Markdown中加入描述文字。這相當於一種**LLM輔助OCR**。然而，就傳統OCR角度，MarkItDown對多語種的支持未明確說明，推測若使用Tesseract則需自行安裝對應語言包。整體來說，MarkItDown的OCR屬於「可以用但不突出」，其設計初心是處理常見格式文件而非專攻掃描件，因此在純文字提取之外的OCR場景下可能不如Marker和MinerU這種專門工具。

- **pdf2md (GPT-4 Vision)：** 這種方法完全依賴GPT-4的視覺理解作OCR。GPT-4作為多模態模型，能夠讀取圖片中的文字並將其轉寫出來，在許多場合下效果不亞於傳統OCR引擎。同時，因為GPT-4本身是多語言訓練的，對英文之外的文字也具有一定識別和理解能力（包括拉丁字母和非拉丁字的語種）。實際測試表明，GPT-4能正確讀出圖片上的英文句子，對簡體中文等也能進行識別和翻譯。不過，其在特殊字體或手寫體上的OCR能力可能較傳統專門OCR差一些。此外，GPT-4視覺目前對每張圖片大小有限制，需要將PDF切頁轉圖片後逐頁發送，不能一次全頁超高解析度。因此，**pdf2md的OCR能力很大程度上取決於GPT-4**：對常規打印體文本，準確率很高；對稀有語種或特殊格式，可能會出現錯漏。值得一提的是，GPT-4不僅識別文字，還**理解圖像內容**，例如對於簽名欄這種手寫簽名，它有可能辨識那是簽名而非正文。然而在Markdown輸出時，GPT-4通常會忽略純署名的塗鴉，不會嘗試去「讀」簽名（避免產生無意義內容）。總體而言，pdf2md在OCR方麵的表現屬於**高水準OCR**（接近人類理解圖像的程度），但使用它需要有OpenAI API權限和資源。

綜上，MinerU在多語種與困難OCR場景上最為強大；Marker緊隨其後，能滿足一般OCR需求；MarkItDown則偏重輔助性OCR；而pdf2md依賴的GPT-4在大多數情況下表現優異，但不可避免受制於API使用成本與一些極端情況限制。

3. 段落斷句與多語翻譯支持

段落斷句（Segmentation）： 將PDF轉換為Markdown時，正確的段落和句子邊界對後續翻譯與摘要尤為重要。如果段落斷行不當（例如每行結尾都產生換行），翻譯時可能造成語意破碎或模型誤讀。所幸這幾款工具在段落處理上都有相應對策：

- Marker 和 MinerU 都有 **去除頁面雜訊與拼接段落**的功能。Marker在提取文本後會進行清洗，比如移除頁眉、頁腳、分欄殘留、分頁符號等，以確保輸出的Markdown文本在語義上連貫 [5]。同時，Marker透過偵測段落塊之間的關係來決定是否插入段落分隔（如額外空行），從而**保持原文段落的連續** [39]。MinerU亦明確提到會去除分頁號碼、重複的頁眉頁腳，使段落順暢連接 [9]。它輸出的內容是依真實閱讀順序排序，而非PDF視覺上的順序 [40]（例如雙欄文本會自動按欄順序串接）。因此這兩者產出的Markdown一般不會出現每行斷句混亂的問題，而是完整的段落段落相接，有利於後續逐段翻譯或摘要。

- MarkItDown 相對簡化處理，因為它**定位就是產生給LLM看的Markdown** [17]。為了降低LLM負擔，它傾向輸出精煉的內容。例如MarkItDown可能直接將連續的文字合併成一個大段落，以減少多餘的格式符號。因此斷句方面，它不會刻意每行加換行符，而是保持原文本的自然段落。但是需要注意，MarkItDown在解析PDF時主要依賴內部heuristics（可能基於版面坐標或換行符判斷）。在某些複雜排版下，可能出現**句子被切開**或**多個段落連成一段**的情況（雖不常見）。整體而言，MarkItDown輸出的Markdown還算乾淨，段落邊界大體正確，適合直接丟給翻譯模型處理。

- pdf2md 由於每頁單獨交給GPT處理，它的 **段落連續性**可能在跨頁時出現問題。也就是說，如果一個段落橫跨兩頁，GPT在處理第一頁時只能看到該頁內容，不知道下一頁其實是同一段的延續。因此pdf2md默認會把頁末文本當作段落結尾，在Markdown輸出加上一個分隔線或斷行（例如示例中每頁結束會有「----------」分隔 [41] [42]）。這意味著長段落可能被硬生生截成兩段，需要人工或後續程序來合併。此外，GPT對頁面

的處理可能略帶主觀，比如它可能在頁面開頭附註「*Page 2*」之類的標記[41]。這些在後續整理時需要移除以重建完整段落。為了解決這問題，一種方法是修改prompt，提示GPT忽略分頁概念，或由腳本在合併各頁Markdown後執行清理。總之，pdf2md能**正確識別頁內**的句子和段落，但跨頁連貫性需要特別處理。

多語翻譯支持： 就這四款工具本身而言，**都沒有內建自動翻譯**功能——它們的職責在於內容提取和格式轉換。然而，它們的輸出非常適合作為後續翻譯的輸入。通常的做法是先用這些工具拿到英文Markdown，再將Markdown內容送入翻譯系統（人工或機器）轉為中文。

- Marker與MinerU因為**完整保留結構**且段落連續，翻譯時可以逐段處理而不怕順序錯亂。例如，可將它們輸出的Markdown拆分成合理長度的段落片段，用ChatGPT或其他翻譯引擎翻譯，再重組回Markdown格式。由於這些工具已提供標題層級和清單縮排，翻譯時只需替換文字部分即可，格式標記留用即可保留原結構。

- MarkItDown的輸出簡潔清爽，直接全篇送入大模型進行翻譯也相對安全，因為沒有太多雜訊標記干擾。微軟設計MarkItDown的目標之一就是**供LLM下游任務使用**[17]，因此在翻譯情境下，它輸出的Markdown是"LLM友好"的。當然，仍需要注意分批翻譯以避免上下文超長和減少幻覺風險。

- pdf2md如果要直接產出翻譯結果，其實**完全可行**：只需修改提示詞，例如將「Convert … to Markdown」改成「Translate … to Chinese Markdown」並調整要求。GPT-4完全可以在閱讀圖像的同時直接給出譯文Markdown。然而，直接讓GPT同時執行識別+翻譯兩個任務，可能會增加出錯幾率（例如某些專有名詞翻譯錯）。為謹慎起見，通常建議還是**分步**：先用GPT-4視覺抽取英文Markdown，再用語言模型翻譯成中文。值得一提的是，有些新型的大模型聲稱有超大上下文（如Anthropic Claude 100k、Gemini Flash 2.5等），理論上可以直接吃下一整本書的Markdown一次翻譯。但實踐中，超長上下文模型的穩定性和翻譯風格仍在觀察中。綜上，這些工具本身不翻譯，但都可與翻譯模型**串接**形成工作流，其中MinerU/Marker產出的高質量結構有助於減少翻譯錯漏，而MarkItDown簡化的輸出有助於模型聚焦內容本身。

4. 大文件與效能表現

處理博士教科書級別的長篇PDF（可能數百頁、數十萬字）對工具的效能是考驗。各工具在大文件支持上情況如下：

- **Marker：** Marker是用Python實作的，本身會逐頁處理PDF內容並組裝結果。由於它內部採用了高效的pdfium引擎和多線程（workers）處理[43]、以及可選的批次OCR，因此對於**中大型PDF**能較好地擴展。實際部署中，Marker常被架設為一個內部服務，接受PDF輸入後異步返回結果[6]。有用戶反饋Marker對於自己的批量PDF解析需求「相當好用」，可以作為後端服務供多任務使用[44]。然而，當PDF特別大時（比如上千頁），記憶體使用和處理時間會相應增加。如果沒有足夠計算資源，可能需要拆分文檔或分批處理頁面來降低單次負荷。總的來看，Marker在大文件處理上**沒有硬性上限**，但受限於本地硬體；在給定硬體下屬中等（水準）效能。

- **MinerU：** MinerU對硬體要求相對高。官方建議至少16GB RAM，最好32GB以上，並推薦使用GPU加速[45]。因為MinerU用到了深度學習模型（版面分析YOLO、公式識別Transformer等[46][12]），處理每頁開銷較大。但MinerU也在持續優化效能：如重構管線並更好利用資源，使在高配機器上**整體解析速度提升超過50%**[47]。對於大型PDF，MinerU採取分段加載和流式處理策略，可以逐頁解析並寫出結果，減少一次性占用。它也支持在Linux下利用多卡GPU並行處理頁面。換言之，MinerU能**勝任超大文件**的解析，但前提是有足夠硬體支撐。在資源允許下，其速度可以很快；資源不足時處理時間就會拉長甚至可能因記憶體不足而失敗。因此MinerU更適合在服務器環境或本地高性能工作站上運行，用於那些要求高準確度的大批量文檔轉換。

- **MarkItDown：** MarkItDown以輕量級著稱，它對記憶體和CPU的佔用相對較小。這部分是因為它對不同格式檔案使用專門的轉換策略，許多情況下不依賴深度學習模型。例如對PDF，它可能使用pdfplumber或PyMuPDF等純粹解析PDF文本的方法，因此速度很快。然而正如某些開發者指出的，**快速的方法往往只能達到80%的效果**，剩下20%覆蓋所有極端情況則很難 [48] [49]。MarkItDown在面對非常複雜的版面時，可能就只提取純文字而捨棄難以解析的結構（這也使它保持速度和穩定）。因此對超大文件，如果內容較標準（比如書本文字），MarkItDown幾乎可以**無壓力**地從頭到尾轉出Markdown；但如果每頁都有極複雜的表格或圖像，它仍然能跑完整本書，只是結果中那些複雜元素會變成純文字塊，失去原本精細的格式。總體而言，MarkItDown非常適合**快速遍歷整本書**提取文字，處理過程資源友好，很少會因文件過大而崩潰。

- **pdf2md（GPT-4 Vision）：** 理論上，GPT-4可以逐頁處理，再多頁也只是API調用次數的增加。然而在實踐中，有兩個限制：**時間/費用**和**一致性**。一份500頁的教材，如果每頁用GPT-4解析5秒，那總耗時約2500秒（40多分鐘），花費（按現在API定價）可能數十美元以上。這還不包括排隊等待或速率限制的時間。另外，GPT對每頁獨立解析後，需要我們把結果彙總。由於是分頁獨立運算，出現不一致風格或措辭也是可能的（比如同一術語在不同頁GPT可能翻譯或格式化略有差異）。Graphlit公司的測試表明，使用Anthropic的多模態模型（Sonnet 3.5）配合他們的LLM模式，也能很高質量地抽取整份PDF，尤其在表格上勝過傳統方法 [50] [51]；但這同樣涉及高昂的計算成本。因此，pdf2md這類方法對超大文件**可行但代價高昂**。一般建議只在特定需要時針對局部或中型文檔使用，或作為其他工具的補充。若一定要用於全書，可以嘗試降低每頁解析DPI或僅重點頁使用，以平衡速度和成本 [52] [19]。需要注意的是，OpenAI API對併發請求和總tokens都有限制，雖然該腳本允許8頁並行處理 [53]，但超大文件可能仍會因API限制需要**分段多批**完成。

5. 輸出格式與適配性

在輸出格式方面，Markdown是共同點，但各工具處理細節略有不同。另外還有JSON等格式的支持情況：

- **Marker：** 直接生成符合Markdown語法的文本檔，並輸出對應的**JSON**結構（包含版面區塊資訊）。Markdown中會引用圖像檔案（如 `![](image1.png)`），這些圖像檔會另外匯出保存 [22]。Marker的Markdown格式對人類和機器都友好，可用於筆記工具或繼續進行LLM處理。由於連圖片、公式都考慮到了，它適用在需要**完整重構文檔**的場景，比如把一本PDF教材轉成帶圖的Markdown筆記，或發佈到GitHub Pages等。Marker的JSON輸出則方便進一步自動處理，比如開發者可據此分析文檔結構、抽取特定元素等。

- **MinerU：** 也提供Markdown和JSON兩種輸出。值得一提的是，MinerU的Markdown有**「多模態Markdown」**和**「NLP友好Markdown」**之分 [54]。所謂多模態Markdown，指的是帶有圖像（以及表格以HTML表示）的豐富Markdown，適合人閱讀；而NLP Markdown可能是純文字的（例如把圖表內容轉成描述），更適合下游NLP任務。使用者可以依需求選擇。MinerU提取的表格會以HTML表格嵌在Markdown中，這對於像Typora、Obsidian這類支援原生Markdown表格渲染的工具來說可能需要額外轉換（因Markdown本身不支援複雜表格）。不過在GitHub Pages或一般網頁環境下，Markdown中的HTML表格可以正常渲染，因此也是合理的權衡。MinerU還支持輸出**中間格式**，比如帶版面坐標的JSON、排版可視化的結果等 [55]（便於確認解析質量 [56]）。綜合而言，MinerU的輸出選項最為豐富，Markdown適合直接拿來做人類可讀的知識庫文章。

- **MarkItDown：** 輸出純文字形式的Markdown字符串，不會額外產生圖檔等附件。對於文檔中的圖片，MarkItDown預設行為可能是插入一個描述（若有OCR或EXIF資訊）或僅標示圖片存在，但**不會自動提取圖片**成檔案。它更關注文本內容的提取。因此，用MarkItDown轉換後若需要圖像，可能要手動從PDF擷取圖片並插入Markdown。MarkItDown不提供JSON結構輸出，但因為它支援許多格式轉換，在流水線中可靈活運用。例如，可以先用MarkItDown把PDF轉Markdown，再用其他腳本將Markdown轉成HTML或其它格

式。總的來說，MarkItDown的輸出**最簡單乾淨**，非常適合馬上拿去喂給LLM做後續問答或摘要。但若期望富媒體筆記，MarkItDown的輸出需要進一步加工補充。

- **pdf2md（GPT-4 Vision)**：其輸出就是GPT模型直接產生的Markdown文字。這通常**已經包括了圖表描述**等內容。例如，當遇到表格時，它會在Markdown中畫出對應的表格語法 [18] ；遇到超連結時，也會以Markdown格式保留（如`[鏈接文字](URL)`） [42] 。不過對於照片或插圖，GPT不可能生成真正的圖像文件，只能寫出一個描述或留空。實際上，在沒有特別指示下，GPT-4傾向於忽略純裝飾性的圖片，或者僅給出簡短的描述（例如「圖：某某流程圖」）。因此pdf2md拿到的Markdown對於**以文字為主**的內容是完善的，對於依賴圖形的內容就欠缺直觀呈現。如果需要包含原圖，還得人工從PDF中提取圖片並插入Markdown適當位置。由於GPT是直接生Markdown，免去了我們自己處理格式的麻煩，這對很多應用非常方便。此外，pdf2md沒有JSON輸出，但我們可以把GPT的Markdown再轉換為需要的格式（比如HTML、PDF等）作其他用途。

適配多平台： 最終的Markdown通常會被放入Typora、Notion、Obsidian、Medium或GitHub Pages等平台。上述工具產出的Markdown基本都遵循標準語法，兼容大多數平台。但還是有幾點需要注意： - 若平台不支援LaTeX數學公式渲染（例如某些部落格系統），Marker和MinerU輸出的公式可能需要轉為圖片或使用插件支持。 - MinerU的HTML表格在一般Markdown編輯器中可能不預覽渲染，但在網頁發布時沒問題。必要時可考慮將HTML表格轉換為Markdown簡易表格或圖片。 - 帶有大量圖片連結的Markdown（如Marker/MinerU）在Notion等平台上導入時，需確保圖片檔一同上傳，否則連結會失效。 - MarkItDown和pdf2md輸出因為圖像部分缺失或只有描述，在知識筆記場合可能需要補圖，但在Medium等發表純文字內容時倒是剛好減少繁瑣。

總之，各工具輸出的Markdown稍加調整即可用於目標平台。從**完整度**角度看，MinerU和Marker適合需要圖文並茂、公式精確的技術筆記；MarkItDown適合純文字知識庫或作LLM輸入；pdf2md適合高質量文本再創作或內容復用場景。

延伸工具與整合方案探索

除了上述比較的工具外，市面上還有其他開源或商業解決方案值得關注，特別是一些可以將PDF轉Markdown、翻譯和摘要整合成流水線的工具鏈：

- **IBM Docling：** 這是IBM在2024年開源的新庫，可高效解析PDF、DOCX、PPTX等並匯出Markdown/JSON。Docling採用了IBM深度搜尋專案的技術，擅長處理學術論文等複雜文件。它號稱能像Marker/MinerU一樣識別**方程式、圖像、表格**並轉為Markdown或LaTeX [57] 。由於IBM的背景，Docling在企業文檔處理上可能表現出色。實際上，有用戶分享他們成功用Docling替代之前調用外部OCR API的方案，直接在本地完成了OCR+抽取，再配合本地LLM進行分析，感到非常方便 [58] [59] 。目前Docling仍在演進中，但已成為開源社群熱門專案，其易用性和多格式支持使其成為Marker/MinerU之外的有力競爭者。

- **AllenNLP olmOCR：** 這是一個來自Allen Institute的前沿研究項目，目標是利用**視覺語言模型**高效提取PDF文字。olmOCR被稱為“PDF線性化”工具，即將PDF內容轉成線性可讀的文本序列，主要用於為LLM提供海量訓練資料 [60] 。據論文報導，olmOCR在一些困難場景下的輸出質量**顯著優於**其他工具（Marker、MinerU、GOT-OCR等），在ELO評分中超過1800分 [61] [62] 。這表示olmOCR在準確還原方面可能設定了新的標杆。不過，也有測試者反映，在某些PDF上olmOCR的錯誤反而比MinerU更多 [63] ——可見這類模型仍在不斷改進。olmOCR目前開源提供了工具包 [64] 和在線Demo，可以嘗試對比。由於它專注於**高品質文字提取**，未必直接給Markdown（可能輸出純文本或帶標註的文本），但我們可以把它看作MinerU這類傳統流水線方案的補充或未來方向。

- **Unstructured.io 工具集：** Unstructured是一組用於文件解析的Python庫，強調將文件劃分為結構化「元素」(elements)，如標題、段落、表格等。它對PDF、HTML、電子郵件等都有支援，在RAG（檢索增強生成）應用中常被用來預處理文檔。雖然Unstructured本身不直接輸出Markdown，但開發者可以輕鬆地將解析得到的元素重新組裝成Markdown格式。它的優點是**模組化**且社群活躍，不斷有新解析器插件加入。缺點是對非常複雜的布局可能需要調參。另外，Unstructured更適合拿來做**知識庫構建**中的解析步驟，然後配合向量資料庫和LLM做問答，而非單純為了生成美觀Markdown。

- **商業API服務：** Tech巨頭們提供的文件AI服務也值得一提：

- *Microsoft Azure Form Recognizer / Document Intelligence*：能將文件解析成結構化JSON，包括段落、表格（帶單元格坐標內容）、發票欄位等專業抽取 [33]。之前有用戶比較Azure與Marker，認為Azure的結構識別（特別是表格）質量更高，但代價是商業API成本 [34]。不過Azure提供預建模型和自訂訓練選項，適合企業批量處理或針對特定模板優化。在拿到JSON後，我們可以轉為Markdown或直接讓LLM利用JSON進行說明生成。
- *Google Cloud Document AI*：類似Azure服務，也能高品質提取文檔文字、段落、表格結構和版面信息，並且對PDF中的手寫和印刷字都有專門模型。Google的優勢在於其OCR技術積累和對表單、表格類型文件有專精的parser。用它轉換教材這類自由版面內容也是可以的，只是沒有專門為Markdown輸出設計，需要自行後處理。
- *Amazon Textract*：亞馬遜的文件提取服務，強調可靠性和與AWS生態整合。它能辨識文本、表格和關鍵-值對（例如表單欄位）。對一般書籍的解析也沒問題，但和Azure/Google一樣，輸出為JSON結構，需要自定義轉Markdown。
- *Mathpix*：這是一家專注於**科技文檔OCR**的公司。Mathpix的旗艦產品可以把PDF轉為LaTeX、Markdown等，特別擅長識別**數學公式**和生成對應的LaTeX標記。社群反映Mathpix效果非常驚人，「PDF中的方程、圖片、表格通通轉成Markdown/LaTeX，幾乎毫無錯誤」 [57]。Mathpix提供有限度的免費使用，超出後按頁計費。對於需要轉換含大量公式的論文論著，Mathpix是商業解決方案中的翹楚。

- 其他如*ABBYY FineReader SDK*、*Adobe PDF Extract API*等傳統廠商的技術，在精度和結構保留上也很強，但是通常沒有直接Markdown輸出，需自行格式化整合。

- **Pipeline 與 Agent 框架：** 若希望**自動化整合**PDF轉換、翻譯、摘要整個流程，可以考慮使用一些管線工具或Agent框架：

- *LangChain* 等流水線框架：可以串聯不同階段。例如先調用MinerU API轉Markdown，再調用翻譯模型API，最後調用總結模型，把結果組裝成所需格式。LangChain容許插入自定義工具，因此上述開源工具都可包裝成一個步驟。這種做法的優點是**靈活**：可以選擇最佳轉換工具+最佳翻譯引擎+最佳摘要模型的組合。缺點是要寫組裝代碼，並處理各步的輸入輸出格式對齊。
- *Hugging Face Pipelines*：利用Transformers提供的管線，可以在本地進行一些步驟（如OCR、翻譯）。例如用LayoutLM等模型先做文檔結構預測，再用MarianMT或M2M100模型翻譯，最後用T5或GPT-3.5生成摘要。這需要一定機器學習專業知識才能調優。
- *PDF-Extract-Kit* + 自訂Stage：值得一提的是OpenDataLab除了MinerU，還推出了**PDF-Extract-Kit**框架，它採用模組化Stage設計，允許開發者插入自訂處理步驟 [65] [66]。官方就建議開發者可基於PDF-Extract-Kit打造自己的應用，如**文件翻譯、問答**或助手系統 [67]。也就是說，可以把PDF-Extract-Kit的解析結果交給一個翻譯Stage（裡面調用翻譯API），再交給摘要Stage，由此構建端對端的自動處理鏈。在未來，他們可能直接提供一些樣板專案供參考 [68]。

- *Graphlit* 平台：Graphlit是一個新興商業平台，強調用LLM處理文檔。他們比較了多種方案，聲稱自己結合Anthropic多模態模型的Markdown提取在表格上效果最佳 [50] 。這類平台其實就是把上面的步驟封裝起來，提供一站式服務（上傳PDF→下載Markdown/摘要），適合不想自己折騰整合的團隊。

綜合來看，**開源方案**方面Marker、MinerU、Docling、olmOCR各有千秋，可以根據具體需求混搭使用；**商業方案**提供了更高穩定性和一些獨有能力（如對某類文件特化的提取），但成本較高且靈活度有限（黑箱服務）。而藉助流水線/Agent框架，最終我們有機會打造一個將上述功能串聯的自動化系統：從PDF內容擷取到雙語資料產出，再到知識萃取與應用，全部無縫連接。

未來發展方向：Agentic AI與多模態智慧文檔處理

展望未來，PDF等文件的智慧處理將更深入地結合**Agentic AI**和**多模態理解**技術。我們可以想像一個智能代理，具備視覺和語言雙重能力，能自主地對輸入文件做出決策式的解析：

- **動態OCR與版面分析：** 未來的系統將能自適應選擇處理策略。例如當代理打開一份文件時，先經過版面分析模型判斷頁面類型：是純文字頁、表格報表頁、表單頁，還是含有照片/簽名的頁？根據判斷結果，代理會選擇不同工具。比如文字頁用傳統解析提取文字，表格頁調用專門的表格識別模組（如深度學習表格結構模型），表單頁則提取關鍵欄位，帶有簽名的頁面則啟用圖像識別模型來標記簽名圖像（而不試圖OCR它）。整個過程由Agent協調各模組的輸入輸出，最後組裝回完整的結果。這種**自適應管線**超越了單一模型端到端做所有事情的模式，更具靈活性和可控性。

- **多模態大模型融合：** 隨著OpenAI、Google等推出更強大的多模態大型模型，我們預期將來會有模型同時具有人類水準的視覺理解和語言生成能力，甚至能產生圖像。這意味著將來我們把PDF"展示"給AI，它不僅讀出文字，還看懂圖裡的曲線趨勢、表情符號等，然後直接產出帶有解析和**智能註解**的Markdown。例如，遇到供應鏈表格，AI不僅轉成表格Markdown，還能附帶一段分析該表格的結論；遇到簽名欄，AI會標記「這裡是簽名：可能是某某的簽名」，甚至可以將簽名圖像存檔供驗證。要實現這些，需要訓練AI在提取事實的同時做**語義分類和理解**。目前一些研究（如Microsoft LayoutLM系列、PaLM-E等）正朝這方向努力，把視覺和語義嵌入融合，用於文檔問答與信息抽取。

- **代理協同和自我檢查：** Agentic AI的另一特點是可以通過對話式反饋來完善結果。未來系統或許會讓AI代理先初步轉換文件，再啟動第二個代理對結果做審核。例如第二代理會校驗數字是否有漏提取、表格欄數是否匹配、翻譯是否保留專有名詞。兩個代理彼此對話，發現問題就回查原PDF或網絡資料修正。這種**多代理協作**將大幅提高最終輸出的可靠性，避免人工去校稿。

- **更廣泛的非結構資料處理：** 除了傳統紙質文件，以後的多模態系統還可以處理**任意視覺+文本混合的資訊**，比如截圖、照片、白板筆記等。一個Agentic AI完全可以被賦予任務："幫我把這場會議白板上的內容做成Markdown總結並翻譯成中文"。它會先OCR白板上的字，辨識圖示，然後組織成文本，加上自己的理解形成條理分明的筆記風格Markdown。同理，各種非標準化PDF（例如帶藝術字的宣傳冊、文字與圖形混排的簡報PDF）都可在多模態模型的輔助下變得井然有序。

總的來說，未來發展將走向**人機協作**的模式：以AI代理為核心，結合專業模型與工具，處理文件中不同模態的訊息，並最終產出生動且有深度的結果。對於像博士教科書這樣的信息密集型文件，不僅可以轉成Markdown文本，還能萃取出知識圖譜、生成教學PPT、提出考試題目等更高層次的產出。屆時，目前困擾的幻覺問題也會因為AI深度理解而減少，翻譯將更貼切專業，摘要也更可靠。

最佳流程建議與結論

綜合以上比較，我們建議針對**英文專業教材**的轉換任務採用以下最佳實踐流程：

1. **PDF 轉 Markdown：** 使用 MinerU 或 Marker 將PDF內容高保真地轉為Markdown（兩者擇一即可）。MinerU適合含大量公式或特殊版面的文檔，Marker則勝在部署簡便和速度尚可。這一步將產生結構清晰的英文Markdown檔，其中包含所有文本、表格（轉為Markdown/HTML）、圖片（已匯出）、公式（LaTeX）等。
2. **文字內容翻譯：** 將第一步得到的Markdown文本進行翻譯，可選擇專業機器翻譯API（如DeepL、Google翻譯企業版）或使用大型模型。在避免幻覺方面，建議**逐段翻譯**而非整篇同時翻。可以編寫腳本將Markdown按段落切分，送入模型翻譯成中文後再拼接。同時保持Markdown語法不變（僅翻譯純文字部分）。這樣可確保格式完整，減少模型擅自改動結構的可能。若使用ChatGPT進行翻譯，應明確提示模型忠實翻譯不要省略或添加。對於Gemini Flash這類超長上下文模型，可嘗試一次讀入全篇Markdown讓它平鋪直敘翻譯，然後由人工或校對程序檢查專有名詞和措辭。

3. **內容結構化與摘要：** 有了中文Markdown後，可進一步利用LLM進行內容再組織。例如讓ChatGPT分析Markdown生成**條列式重點摘要**、**章節概要**，或者提取關鍵詞、製作知識地圖。如果要編寫教學roadmap，可以要求模型根據教材章節順序羅列學習步驟和重點難點。這一步驟因為有前面準確的資料做基礎，模型發揮創意的同時幻覺風險會降低。此外，我們可以把翻譯後的Markdown導入Obsidian等，手工整理小量格式，增補一些註解，再從中產出博客文章或教學講義。

4. **質量檢查與迭代：** 最後對譯文和摘要進行人工審閱是必要的。可利用雙語對照的方式，比較英文Markdown與中文Markdown，確保無段落遺漏。對模型生成的摘要，核對其論述是否忠於原文。如果發現偏差，可將問題反饋給模型讓其調整（例如提示它某段有誤請改正），或者手動修訂。透過幾次迭代，最終獲得高質量的雙語筆記和總結。

這一流程相當於將精確的提取工具與強大的生成模型優勢結合：**先抽取，後生成**，既保留了內容的真實可靠，又充分利用AI加速整理與翻譯。在目前技術水平下，這是比單純依賴一種模型更穩健的策略。

總而言之，PDF 轉 Markdown 再到翻譯和知識重組，是一個需要多階段配合的複雜任務。透過我們對Marker、MinerU、MarkItDown、pdf2md等工具的比較，可以看出沒有單一完美解決方案，而是各有側重。我們應根據具體需求（如對結構完整性的要求、預算和硬體、處理文本種類等）選擇合適的工具或組合。同時關注新興的方案（如Docling、olmOCR）和商業服務，在特殊情境下加以利用。隨著AI技術的發展，未來我們有望看到更加智能的一體化系統，讓從PDF知識到Markdown內容的轉換變得幾乎無縫且高質量。屆時，不論是撰寫學術筆記還是製作教學內容，繁瑣的格式轉換都將由AI自動完成，人類則能專注於創造和理解內容本身，極大提升生產力。

參考資料：

- Marker 專案 (VikParuchuri/marker) 的功能與架構 20 69
- MinerU 專案 (opendatalab/MinerU) 的特性說明 9 11
- Microsoft MarkItDown 的README與使用反饋 17 15
- zyocum/pdf2md 專案說明與範例 18 70
- Hacker News與Reddit上關於工具表現的討論 1 50
- 相關比較文章和報導 71 67

1 2 6 23 33 34 44 Have you tried https://github.com/VikParuchuri/marker ? | Hacker News
https://news.ycombinator.com/item?id=41106888

3 4 5 7 8 20 21 22 30 31 32 35 36 39 43 69 VikParuchuri/marker. Continue this conversation at http://localhost:3000?gist=eebeb4250a5f0774f7717d9d234bd7d4 · GitHub
https://gist.github.com/m0o0scar/eebeb4250a5f0774f7717d9d234bd7d4

9 10 11 12 13 14 24 25 26 37 40 45 46 47 54 55 56 65 66 GitHub - opendatalab/MinerU: A high-quality tool for convert PDF to Markdown and JSON.一站式开源高质量数据提取工具，将PDF转换成Markdown和JSON格式。
https://github.com/opendatalab/MinerU

15 48 49 MarkItDown: Python tool for converting files and office documents to Markdown | Hacker News
https://news.ycombinator.com/item?id=42410803

16 27 Deep Dive into Microsoft MarkItDown - DEV Community
https://dev.to/leapcell/deep-dive-into-microsoft-markitdown-4if5

17 28 29 38 GitHub - microsoft/markitdown: Python tool for converting files and office documents to Markdown.
https://github.com/microsoft/markitdown

18 19 41 42 52 53 70 GitHub - zyocum/pdf2md: Convert PDF to Markdown via OpenAI multi-modal text/vision model.
https://github.com/zyocum/pdf2md

50 51 Comparing the latest API services for PDF extraction to Markdown : r/Rag
https://www.reddit.com/r/Rag/comments/1g6p069/comparing_the_latest_api_services_for_pdf/

57 58 59 Docling is a new library from IBM that efficiently parses PDF, DOCX, and PPTX and exports them to Markdown and JSON. : r/LocalLLaMA
https://www.reddit.com/r/LocalLLaMA/comments/1ghbmoq/docling_is_a_new_library_from_ibm_that/

60 Efficient PDF Text Extraction with Vision Language Models - olmOCR
https://olmocr.allenai.org/blog

61 Papers Explained 326: olmOCR - Ritvik Rastogi - Medium
https://ritvik19.medium.com/papers-explained-326-olmocr-bc9158752901

62 71 olmOCR: Revolutionizing PDF Text Extraction with AI | AGIEntry News
https://agientry.com/blog/375

63 Could you evaluate with MinerU, GOT-OCR, olmoOCR, MarkItDown?
https://github.com/VikParuchuri/marker/issues/587

64 allenai/olmocr: Toolkit for linearizing PDFs for LLM datasets/training
https://github.com/allenai/olmocr

67 68 GitHub - opendatalab/PDF-Extract-Kit: A Comprehensive Toolkit for High-Quality PDF Content Extraction
https://github.com/opendatalab/PDF-Extract-Kit