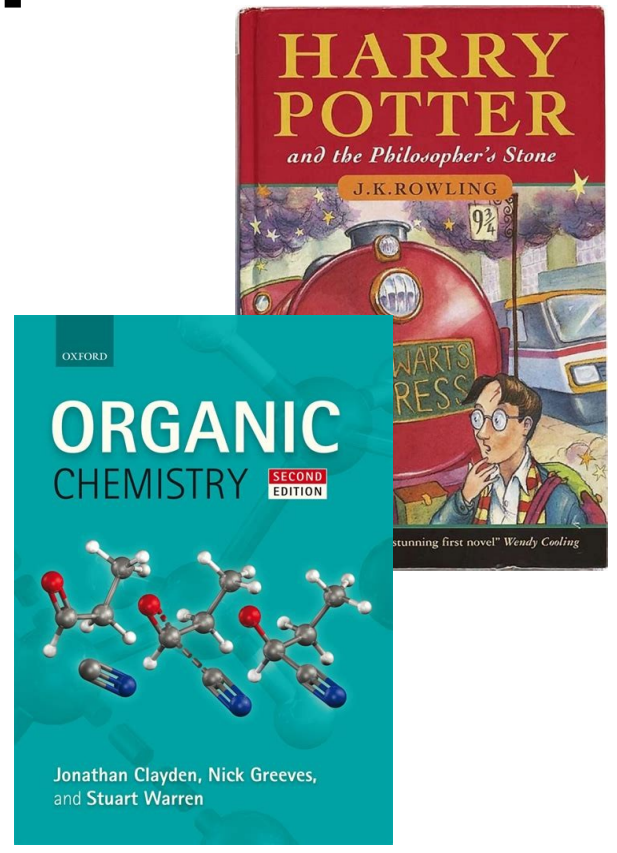
ReadPilot

Your copilot for Documents on MS Edge

By Sambhav Mattoo

Motivation

- Talking to an Ebook?
- Context and detail.
- Precise data extraction.
- Use cases: Recreational, Educational, Legal etc.



Technical Challenges

- Large documents cannot fit into LLM context lengths!
- Vectorization takes time and space!
- 500 pages – Multiple MB – 3-5 minutes
- Smarter approach needed!

Read like a Human

- We don’t read the whole book to quote it!
- We find the index or assume one!
- We leaf through the book to know what’s where!
- We pay attention to the right part!

ReadPilot - Introduction

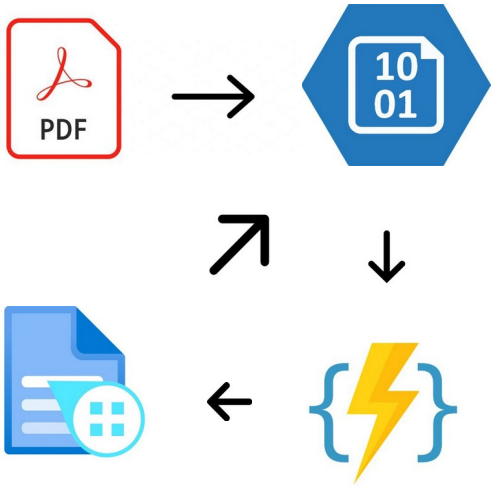
- Copilot app for MS Edge (chromium).
- Modern, minimalist UX.
- Azure backend.
- Precise, with quotations!

System Design - Frontend

- Chromium extension.
- Simple chat interface with linking.
- PDF.js web app to go page.
- Sends .pdf and user queries, gets answer and references

System Design - Backend

- .pdf goes to Azure Blob Storage.
- Azure Function Apps to run scripts.
- Document intelligence used to extract content.



System Design - Backend

- Index is found else assumed.
- Knowledge map created.
- Number, name, start, end.
- Random sampling to find approximate content.

```python
# Generates a knowledge map (chapter summaries) for the document using GPT-4
def generate_knowledge_map(sections, pages, gpt_client):
    """
    For each section, sample up to 3 pages, concatenate their text, and ask
    Returns a list of dicts: [{chapter_name, start_page, end_page, summary}]
    Args:
        sections: List of section dicts (with start_page, end_page, title)
        pages: List of page texts
        gpt_client: AzureOpenAI client for GPT-4
    Returns:
        List of knowledge map entries
    """
```

```python
# Detects a table of contents (TOC) or index in the first N pages of a document
def detect_index(pages, max_toc_pages=50):
    """
    Scans the first max_toc_pages for a table of contents (TOC) or index.
    Handles multi-page indices by aggregating all detected entries.
    Args:
        pages: List of page texts (strings)
        max_toc_pages: Number of pages to scan for TOC/index
    Returns:
        List of index entries: [{title, page_ref, line, toc_page}]
    """
```

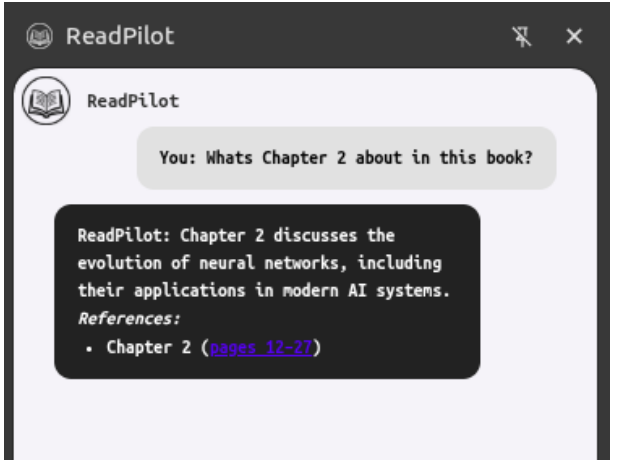
```python
# Segments the document into sections/chapters using the index if found, otherwise creates synthetic sections
def segment_document(pages, index):
    """
    Segments the document into sections/chapters using the index if found, otherwise creates synthetic sections.
    Args:
        pages: List of page texts
        index: List of index entries
    Returns:
        List of sections: [{title, start_page, end_page}]
    """
```

System Design - Backend

- Query + KMap used to score sections using Azure OpenAI.
- Relevant sections vectorized (Azure OpenAI embeddings + Azure AI Search).
- Smart chunking based on Azure Document Intelligence groupings

System Design - Backend

- Azure OpenAI builds final answer.
- Sends reference to chapters used.
- Smooth UX displays with PDF.js page linking.



Solution Performance: Demo

Innovation

- Query very large documents precisely.
- Millisecond latency instead of minutes.
- Vectorize what you need.
- Think like a human. Novel.
- Can be integrated into larger copilot. ;)

Responsible AI

- Quotes where the answer is taken from.
- Secure data storage on Blob, wiped after use.
- No data collection, logging or telemetry.
- Screen reader and keyboard friendly.
- OpenAI content moderation.

Future Improvements

- Better KMapping.
- Adaptive chunking for vectorization.
- Image descriptions.
- Web search agents.
- Integrate with other copilots. ;)

Thanks! uWu