BookBot: Autonomous AI Research and Library System

Overview

BookBot is an AI-driven research assistant designed to curate, manage, and analyze a vast library of books and research papers. It functions as a conversational AI interface that can be asked questions about the contents of books or papers, returning direct citations and detailed answers. The system runs locally on macOS (M4 Mac Mini) and uses the Venice.ai API for large language model (LLM) inference. By operating on a local database of full-text books, BookBot follows a *retrieval-augmented generation* approach: it grounds the LLM's responses on actual content retrieved from its library, ensuring up-to-date and accurate answers. This aligns with the emerging shift in AI from passive assistance (like summarizing a document) to **agentic behavior** – actively searching for and organizing relevant information as part of answering user queries

microsoft.com
. In essence, BookBot aims to be a continuously learning research librarian that **goes beyond search**, providing users with deep insights and exact quotes from their personal digital library.

Key Features

- **Multi-Format Support**: BookBot can ingest and parse a wide range of document formats – from eBooks (EPUB, MOBI) and PDFs to Word documents, HTML, plain text, and more. This ensures that users can include virtually any text-based resource in the library and have it analyzed.
- **Web and macOS Interface**: The assistant is accessible both through a web-based chat interface and a native macOS app (built with Swift). This dual-interface design means you can query your library from a browser or directly on your Mac desktop, whichever is more convenient.
- **Autonomous Research Agent**: BookBot isn't limited to just the books you give it. It continuously monitors global AI research repositories (arXiv, DeepSeek, OpenAI, Meta, Anthropic, Mistral, etc.) to discover new papers and publications relevant to AI development. When it identifies a high-value document (for example, a breakthrough paper on multi-agent systems or a new book on machine learning), it can autonomously fetch and incorporate that text into the library. This way, your knowledge base stays current without manual updates.
- **Calibre Integration**: The system seamlessly integrates with Calibre (the popular open-source e-book management tool). BookBot operates as an invisible enhancement

to Calibre's normal functionality. Users can continue to use Calibre to organize and browse books, while BookBot works in the background to index content, enrich metadata, and generate summaries. This integration realizes a vision long desired by Calibre users – an AI extension that can access the full text of your library and allow you to ask questions about any book via an AI interface.

- **Multi-Agent System**: Under the hood, BookBot is powered by a collection of specialized AI agents, each with distinct responsibilities:
    - **Book Selection Agent** – Scans the library and external sources to identify and prioritize valuable books/papers for inclusion, especially those that will enhance BookBot's ability to answer complex AI-related questions.
    - **Summarization Agent** – Processes books to generate hierarchical summaries. It creates concise book-level summaries as well as chapter-by-chapter outlines. This *hierarchical summarization* approach (first summarizing chapters, then summarizing the summaries) ensures that even very large books can be distilled effectively. The result is a multi-layer index of each book's content, allowing BookBot to quickly retrieve relevant sections when answering questions.
    - **Librarian Agent** – Manages the local database and metadata. It keeps track of what books are in the library, what topics they cover, and where important information can be found. It continuously updates metadata (like tagging books with topics or linking related books) to improve search relevance. This agent also handles *quote extraction and verification*, meaning it can pull the exact wording from a book when providing an answer, ensuring the AI's responses include accurate, verbatim citations rather than hallucinated quotes.
    - **Query Handler Agent** – Acts as the interface during a conversation. When a user asks a question, this agent determines which book or paper (or which chapters) are most likely to contain the answer. It then retrieves the relevant excerpts or summary points and presents them as context to the LLM before formulating an answer. By dynamically activating only the most relevant books for each query, the system keeps the context focused and efficient. This design prevents the need to load an entire library into memory for every question – instead, it's a targeted retrieval of just the pertinent knowledge.
- **Local Knowledge Database**: All books and research papers are stored in a local database enriched with metadata and summaries. This database is structured to allow **fast semantic searches** and direct lookups. For example, you can ask, "Which book explains the concept of hierarchical summarization?" and BookBot will quickly scan its indexed summaries and metadata to find the right sources. Because the data is local, queries are private and fast, and BookBot's knowledge isn't limited by an online index. The stored content, along with the multi-level summaries, often expands the database to several times the size of the raw text, but this trade-off yields highly efficient retrieval.
- **Direct Citation and Verification**: When BookBot provides an answer, it can include citations that reference the exact book and even the page or section from where the information was taken. This is crucial for research and fact-checking. Instead of just saying *"According to **Deep Learning** by Goodfellow et al., XYZ is the case,"* BookBot will provide the exact quote and a reference link or notation to the source material. This

feature turns BookBot into a powerful tool for writing papers or reports, as you can gather authoritative quotes on a topic in seconds. (Under the hood, this is achieved by the Librarian and Query Handler agents working together – extracting the relevant paragraph from the source and supplying it to the LLM so that the answer can be given in the context of an exact reference.)

Architecture

BookBot's architecture combines local computing resources with cloud-based AI inference to create a scalable and responsive system. Here are the core components of its design:

- **LLM Hosting**: The heavy lifting of natural language understanding and generation is done via the Venice.ai API, which provides access to advanced language models. This ensures that BookBot can generate high-quality responses and summaries. By leveraging an API, the local system doesn't need to run a giant model itself, which keeps performance snappy on a Mac Mini while still benefitting from powerful AI computation in the cloud.
- **Local Knowledge Database**: At the heart of BookBot is a local database that stores all the ingested books and documents. This database isn't just raw text; it's *augmented with metadata*. For each book, the database might store the table of contents, chapter summaries, key themes, author information, and more. The text is indexed so that BookBot can perform both keyword searches and semantic similarity searches. In other words, the system can find relevant information even if the user's question is phrased differently than the text in the book. The database is engineered for speed, allowing for real-time query scanning even as it grows to thousands of books.
- **Calibre Plugin System**: BookBot hooks into Calibre's open-source library management ecosystem. This is done through a custom plugin or a direct database integration. When new books are added via Calibre, BookBot detects them, ingests their content, and updates its indexes and summaries. Conversely, BookBot can add metadata (like tags or comments) in Calibre to enrich the user's normal view of their library. This two-way integration means that whether you're managing your books manually or letting BookBot expand the collection, everything stays organized in one place.
- **Dynamic Book Activation**: Rather than loading every book into the AI model's context (which would be infeasible), BookBot uses a strategy of dynamic activation. When a query comes in, the Query Handler agent consults the metadata and summary index to pick a handful of books (or even specific chapters) that are most relevant. Only these texts are then retrieved and fed into the LLM's context window for analysis. For example, if you ask about "hierarchical summarization techniques," BookBot might pull in the summary of a textbook chapter on summarization and an excerpt from an arXiv paper on hierarchical methods. This targeted approach keeps the context window small and relevant, enabling the LLM to focus and provide a more accurate answer.
- **First-Principles Filtering**: BookBot applies a *first-principles* approach to ensure relevant book selection. This means it relies on the structured metadata and content of books to decide what's relevant, rather than black-box model guesses. For instance, if a book's

metadata indicates it's about *natural language processing* and another is about *operating systems*, and you ask a question on AI summarization, the system will exclude the operating systems book from consideration right away. This filtering is done through rule-based checks on categories, keywords, and other metadata. By narrowing the candidate set of books through straightforward logic before doing any heavy AI processing, BookBot saves time and computation. Only the most pertinent texts are considered in the expensive inference step.

All these components work in concert to deliver a smooth experience. The architecture is built to be **modular and extensible** – new agents can be added (for example, a *Translation Agent* to handle multilingual libraries) without overhauling the whole system, since each agent has a clear scope. This design echoes the principles of systems like Microsoft's Magentic-One, where a lead Orchestrator agent delegates tasks to specialized helper agents; such a division of skills into separate agents simplifies development and makes the system more flexible

microsoft.com
.

Relation to CompyMac

BookBot is a **sister project** to **CompyMac**, an AI-driven macOS automation assistant. While BookBot focuses on knowledge curation and question-answering, **CompyMac** is designed to *autonomously control the computer* and execute tasks on the user's behalf (for example, opening applications, managing files, or orchestrating complex workflows on macOS). The two systems complement each other:

- **CompyMac** can call on BookBot whenever it needs in-depth information or research. For instance, if CompyMac is tasked with drafting an email or a report about the latest AI trends, it could query BookBot for content and citations from relevant books and papers.
- **BookBot** benefits from CompyMac's automation capabilities. If BookBot identifies a new important paper on arXiv, it could ask CompyMac to download the PDF and add it to the library without human intervention.
- Together, they form a symbiotic relationship: CompyMac handles *action and execution* in the operating system, and BookBot handles *knowledge and information*. This partnership could enable very powerful autonomous workflows. Imagine asking CompyMac to "Summarize my research and prepare a presentation," and CompyMac uses BookBot to gather the research insights (with citations) and then automatically creates a Keynote presentation for you.

In short, BookBot serves as the **brain and memory** for CompyMac's **hands and eyes**. This division of labor allows each system to specialize and excel at its role, whether it's controlling the computer or providing the knowledge to inform those actions.

Development Goals

To realize the vision of BookBot, the development is focused on a few major goals:

1. **Develop the Multi-Agent Framework**: Implement the core **agent architecture** – including the Book Selection, Summarization, Librarian, and Query Handler agents. Each agent will be built and tested in isolation first, then integrated under a central orchestrating process. This involves designing their communication protocols (what information they pass to each other) and ensuring they can work in parallel where possible.
2. **Implement Global Research Retrieval**: Create connectors or use APIs for external research sources. For example, use the arXiv API to search for the latest papers by keyword, or use OpenAI's or Meta's repositories to find newly published AI research. The goal is for the Book Selection Agent to be able to run on a schedule (say, nightly) to find and suggest new content. This also involves setting criteria for "high-value" – e.g., if a paper is highly cited or a book is recommended by many experts, then prioritize it.
3. **Optimize Book Summarization Pipeline**: Develop an efficient summarization workflow for books. Likely, this will involve using the LLM to summarize text in chunks (e.g. per chapter) and then summarizing those summaries to get a book-level synopsis. We might experiment with different approaches, such as *hierarchical summarization* (which tends to maintain coherence by merging summaries) versus *iterative refinement* (updating a running summary as you go through the text). The end goal is a set of tools that can take a new book and produce a useful multi-level summary and index with minimal human intervention.
4. **Create a High-Speed Local DB**: Engineering the database for speed and scale. As the library grows, naive search will slow down, so we plan to use efficient indices (like keyword inverted indices combined with vector similarity indices for semantic search). We will also need to handle the storage of a lot of text. Part of this goal is deciding on a format: possibly a combination of an SQL database for metadata and a vector store (like FAISS or similar) for embeddings. Testing will involve ensuring that even if we have, say, 10,000 books, a query can be processed in under a second by narrowing down to a few candidate books.
5. **Integrate with Calibre**: Develop a plugin or direct integration with Calibre's database. The objective is to let BookBot import books from Calibre automatically, and also push metadata (like tags or comments) back to Calibre. We'll likely use Calibre's API or a direct database connection to achieve this. A key part of this goal is *seamlessness* – the user shouldn't have to do anything special. For example, if the user deletes a book in Calibre, BookBot should notice and remove it from its index; if BookBot adds a new book, it should show up in Calibre's interface.
6. **Design a Conversational Interface**: Create a user-friendly chat interface for BookBot. This involves front-end design (either web UI or macOS app UI or both) where users can type questions and see answers with citations. We'll incorporate features like showing which source (book/page) an answer came from, follow-up question suggestions, and possibly the ability to click a citation to open that book to the relevant section (if the book is available). The interface should handle multi-turn conversations, maintaining context as needed (with the Query Handler agent managing the conversation state).

7. **Develop Query-Book Matching**: Refine the logic that matches user queries to the right book context. This will likely use a combination of keyword matching (if a book has a chapter titled “Neural Networks,” it’s likely relevant to a question about neural networks) and embedding-based semantic similarity (to catch relevant content where keywords don’t match exactly). The outcome should be that for any given question, BookBot identifies a small set of books or documents and pulls only the necessary excerpts or summaries from them to construct an answer. This step is critical for both **accuracy** (making sure the answer is drawn from authoritative sources) and **efficiency** (not overloading the LLM with too much irrelevant text).

Each of these development goals will be tracked and iterated upon in the repository. As we progress, tests will be written (for example, to ensure summarization doesn’t hallucinate or that query matching has high recall for known question-answer pairs). The final system should meet all these goals to provide a robust, autonomous research assistant experience.

Deliverables

By the end of the development cycle, we expect to deliver:

- **BookBot Prototype (macOS)**: A fully functional local application that runs on macOS (optimized for Apple Silicon). This will include the background services (the agents and the database) and the user-facing interfaces (web and/or native app). Users should be able to install BookBot, point it to their Calibre library, and start asking questions.
- **Autonomous Multi-Agent System**: The implemented and tested multi-agent framework where each agent (Selection, Summarization, Librarian, Query Handler) works as intended. This includes demonstrating that the system can autonomously find a new paper, summarize it, add it to the library, and make it available for queries without any manual steps – a full *end-to-end autonomous pipeline* for knowledge ingestion and retrieval.
- **Enriched Local Library Database**: A database (and likely a set of files) that contains the full text of all ingested books, plus metadata and generated summaries. This database will be several times larger than the original texts due to the added information (which is expected, as mentioned). We will also provide tools or documentation on how this data is stored and how it can be maintained or backed up by the user.
- **Intelligent Query Handling**: A demonstration (and tests) showing that for a variety of user questions, BookBot successfully picks relevant sources and provides answers with correct citations. For example, if asked a question about a concept covered in *three* different books, BookBot might aggregate the viewpoints from those books and cite each one. The deliverable is essentially the assurance (via examples and maybe a benchmarking script) that the query->context selection->answer pipeline is working well and can be easily extended as the library grows.

All deliverables will be documented in the repository. The prototype will come with instructions for installation and use. The code for the agents and other components will be modular to

encourage community contributions or adaptation for other uses (for instance, someone might want to use the Summarization Agent on their own texts as a standalone tool).

Future Expansion

Looking beyond the initial prototype, there are several exciting directions in which BookBot can grow:

- **Deeper CompyMac Integration**: Once both CompyMac and BookBot are mature, they could be more tightly coupled. For example, BookBot could function as a plugin within CompyMac, so that any time CompyMac encounters a task needing information (like "find me the definition of X" or "research Y and generate a summary"), it automatically leverages BookBot. This could turn CompyMac into a super-charged personal assistant that not only performs actions but also informs its actions with deep knowledge.
- **Additional Research Sources**: The initial version focuses on AI-related sources, but the framework could extend to other domains. We might integrate with scientific databases like PubMed for biomedical information, or JSTOR for humanities, etc., depending on user needs. The idea is to make BookBot a general research librarian for any field. Its autonomous agent could periodically update its knowledge from these sources, keeping the library up-to-date in multiple domains.
- **Collaborative Knowledge Base**: Introduce features for users to **annotate and tag** content, which BookBot can then learn from. For instance, if a user flags a certain passage as important or provides a manual summary, BookBot could incorporate that into its metadata. Over time, a community of BookBot users could even share metadata or summaries (privacy permitting) to collectively improve the system's knowledge.
- **Refined Metadata and Ranking**: Continue improving how BookBot categorizes and ranks information. This might involve implementing more advanced natural language understanding to generate semantic labels for books or using network analysis (e.g., if Book A and Book B are often referenced together in answers, link them in the metadata). The goal would be to make the book selection for answering queries even smarter over time, learning from past queries which sources tend to be most useful.
- **Plugin Ecosystem**: We foresee the possibility of BookBot supporting plugins or extensions. For example, a plugin could allow BookBot to output answers in specific formats (like generating a Markdown report of an answer, or creating flashcards from a textbook). Another plugin might enable voice interaction (so you could *ask BookBot verbally* and hear a spoken answer). By designing the system with clear APIs between components, third-party developers could create such add-ons.
- **Scalability and Cloud Sync**: Although BookBot is a local-first system, users might appreciate the ability to sync their library across devices or run BookBot on a server. Future versions might allow an option to host the BookBot database in a personal cloud or NAS, and then connect to it from multiple devices. We will explore ways to do this securely (ensuring private data stays private).