SwordSearcher Librarian Help

The Librarian is an AI-powered research assistant for your SwordSearcher library. You ask a question in plain English; it searches your commentaries, dictionaries, books, and Bible, then assembles an answer with clickable citations that open the original entries in SwordSearcher.

The Librarian is grounded in your library — citations come from the modules you have installed, not from somewhere on the internet. By default, it draws its answers from those sources and uses general knowledge to frame and explain them. How tightly it sticks to the library versus how freely it speaks in its own voice depends on the **prompt strategy** you pick (see *Prompt Strategies* below). The library and its citations are always the anchor; the rest is a dial you can turn.

Requirements

- **SwordSearcher Deluxe 10.0.3.26017 or later.** The Librarian reads its content directly from the Bible, commentary, dictionary, and book modules included with the full Deluxe edition. Earlier versions and the Evaluation edition do not include the modules the Librarian needs.
- **16 GB of RAM and 6+ physical CPU cores is the minimum for local AI.** Below this floor — for example, a 2019-era Surface Laptop with a 4-core ultra-low-voltage CPU and 8 GB of RAM — local models can take many minutes per answer, sometimes long enough that the planning stage appears stuck. The Librarian shows a slow-hardware warning on the welcome page in that case and recommends Search Only or a Claude / OpenAI cloud profile. 16 GB of *dedicated GPU VRAM* (separate from system RAM) unlocks the best local-model tier.
- **Windows 11 (64-bit, x86/x64).** The Librarian is a 64-bit Windows application built for Intel and AMD processors. **Local AI models are not supported on ARM-based Windows PCs** — including Snapdragon X / Copilot+ PCs and earlier Snapdragon-based devices. The Adreno GPU's Vulkan driver fails llama.cpp's compute path, and the ARM cores running x86 code under Prism emulation are far too slow for

usable inference (a single short answer can take more than ten minutes). On ARM hardware, use **Search Only** mode or a **Claude / OpenAI cloud profile** instead — both run on these machines, since neither uses the local llama.cpp runtime.

A discrete GPU is not required, but speeds up local-model responses dramatically. Cloud profiles (Claude, OpenAI) work on any system since the heavy lifting happens on the provider's servers.

Library Coverage — Deluxe Modules Only

For the time being, the Librarian can only search the modules included with **SwordSearcher Deluxe**. The semantic search indexes that make natural-language lookup possible are built ahead of time and shipped with the Librarian itself, one per supported module. Your own user-created modules — and any third-party modules you have added to your library — are **not yet searchable by the Librarian**, because their search indexes haven't been built. SwordSearcher's regular search across your full library is unaffected; this limitation applies only to what the Librarian can find and cite.

Building search indexes for user-created modules is planned for a future release, but is not yet available. In the meantime, anything you ask the Librarian will be answered from the Deluxe library content, even if the answer is also covered in a user module on your machine.

First-Time Setup

Before you can ask a question, you need to choose a model. The Librarian does not ship with one — pick the one that fits your computer.

1. Click the **gear icon** (⚙) at the top of the chat window. This opens **Model Settings**.
2. Click **Download model...**
3. The list of recommended models appears. Each entry shows its size and which hardware tier it suits. A note at the top tells you what the Librarian detected

about your system (RAM and GPU). **Pick the recommended model that fits your hardware.** Higher quantizations (Q8_0 > Q6_K > Q5_K_M > Q3_K_M) give better answers; pick the highest one your machine can run.

4. Click **Download**. The download is several gigabytes and may take a while. You can resume an interrupted download by clicking Download again.
5. When the download finishes, the model appears in the **Profiles** list on the left. Select it and click **OK**. The defaults are good — you do not need to change anything else.

The Librarian will load the model and the status bar will read **Ready**. You can now ask questions.

If you prefer a cloud model

In **Model Settings**, click **Add**, choose **Claude (Anthropic)** or **OpenAI**, paste your API key, and enter the model name. Cloud models give the highest quality answers but require an API key and an internet connection, and you pay per request to the provider.

Privacy: Local vs. Cloud Models

The recommended (downloaded) models are **local**. Everything happens on your computer — your questions, your library content, and the answers never leave your machine. No internet connection is needed once the model is downloaded.

Cloud profiles (Claude, OpenAI, or any other API endpoint you configure) work the opposite way. Each question, along with the relevant library passages the Librarian retrieved to answer it, is **uploaded to the provider over the internet** for processing. Their privacy and data-retention policies apply. Don't use a cloud profile for material you would not want to send to a third-party service.

The model picker in the toolbar always shows which profile is active, so you can tell at a glance whether the next question stays on your machine or goes to the cloud.

Asking a Question

Type your question in the box at the bottom and press **Enter**. Use **Shift+Enter** for a new line.

The Librarian works best with natural questions like:

- *What does Clarke say about Romans 9:15?*
- *Compare what Larkin and Henry say about the seven churches in Revelation.*
- *Find commentary on the burning bush in Exodus 3.*
- *What do the reference works say about justification?*

While the answer is being prepared, you'll see brief status updates (planning, retrieving, then streaming text). Click the **Stop** button to cancel an answer in progress.

Citations

Every claim in the answer is footnoted with the source the Librarian used. You'll see two kinds of links:

- **Source badges** like *[Barnes]* — click to open that entry in SwordSearcher.
- **Verse links** like *Romans 8:28* — click to open the verse in SwordSearcher.

If SwordSearcher is running, the entry opens immediately. If not, SwordSearcher launches and navigates there. The status bar shows whether SwordSearcher is currently connected.

Copying an Answer

Each message has a **copy button** in its top-right corner. Pasting into a SwordSearcher user module preserves the clickable citations.

Start a New Chat for Each New Topic

This matters more than it sounds. The Librarian remembers earlier messages in the current conversation so you can ask follow-ups like *"tell me more about what Henry said"*. But that memory is also a distraction — if you switch to a completely

unrelated topic in the same chat, the earlier conversation can pull the answer in the wrong direction and waste context space.

Rule of thumb: when you're done with one topic and want to ask about something else, click **New chat** in the sidebar. Each conversation should be focused on one subject.

Conversation History

Past conversations are listed in the sidebar on the left, grouped by date. The Librarian saves them automatically.

- **Click** a conversation to reopen it.
- **Right-click** a conversation to **Rename** or **Delete** it.
- The **trash icon** at the top of the sidebar lets you clear conversations older than a day, older than a week, or all of them.

The Toolbar

Above the chat area:

- **Model picker** — shows the active model. Click it to switch to a different profile you've configured.
- **Gear icon** — opens **Model Settings** to add, edit, or download models.
- **Question mark icon** — opens this Help window.

Switching Models

You can keep multiple model profiles configured (a fast local model for quick lookups, a Claude profile for harder questions, and Search Only for raw retrieval).

Switch between them anytime using the model picker in the toolbar — no restart needed.

Switching models does **not** clear your current chat, but the new model starts fresh on its next reply (it sees the conversation history but has not "thought about" the earlier turns the way the previous model did).

Tips for Better Answers

- **Be specific.** *"What does Spurgeon say about Psalm 23?"* gives a sharper answer than *"What about Psalm 23?"*.
- **Name authors** when you want a particular voice. The Librarian recognizes the author names from the modules you have installed.
- **Use real Bible references** in the form *Book Chapter:Verse* (*Romans 9:15*, *1 Samuel 20:1-4*) so the Librarian can scope the search correctly.
- **One topic per chat.** Start a new chat when changing subjects.
- **Verify the sources.** The Librarian tries to be accurate, but the citations are there so you can read the original yourself. Always click through to confirm anything important.

Search Only Mode

If your active profile is **Search Only - No AI**, the input box becomes a semantic search across your library. There is no language model involved — you get a list of the most relevant chunks from your commentaries, dictionaries, and books, ranked by similarity. Each result is clickable and opens in SwordSearcher.

This mode is fast, runs on any computer, and is useful when you want raw search results without an AI's interpretation.

Troubleshooting

"No model profile configured" — Open Model Settings (gear icon) and either download a model or add an API-based profile.

"SwordSearcher: Not running" — Citations still work; SwordSearcher will launch when you click one. For instant clicks, open SwordSearcher first.

Slow responses on a local model — Your model may be too large for your hardware, or it may be running on CPU instead of GPU. Try a smaller quantization (e.g., Q5_K_M instead of Q8_0). The download dialog labels each tier so you can see which fits your system.

The answer ignored part of my question — Try rephrasing more directly, or break it into two questions. Very long or multi-topic questions are harder for the Librarian to answer well; one focused question per turn works best.

A Note on What the Librarian Is and Isn't

The Librarian only knows what's in your SwordSearcher library and what the language model already knew when it was trained. It does not browse the internet, it does not have access to anything outside your installed modules, and it is not infallible. Treat its answers as a starting point for your own study — the citations are the real product. Read the sources.

Prompt Strategies

Each model profile has a **Prompt strategy** setting that controls how strictly the Librarian sticks to your library versus how freely it can use the model's own knowledge to frame an answer. Three strategies are available; pick the one that matches the model you're using.

Grounded. The strictest strategy. Every claim in the answer must come from a cited source — if the library doesn't cover the topic, the Librarian says so and stops.

General knowledge is not used to fill gaps. Best for smaller local models, where a tight leash keeps them from drifting into invented quotes or made-up authors. Choose this when you want the Librarian to behave like a pure research-assistant: pull the books off the shelf and present what's there, nothing more.

Balanced. The default for local models. The Librarian leads with a direct answer to your question and then supports it with cited sources. General knowledge is permitted for definitions, background, and context, but anything *interpretive* or *theological* must come from the library. This is a good middle ground — answers feel less robotic than Grounded but the substance is still anchored to the sources.

Conversational. The default for cloud models (Claude, OpenAI). The Librarian is trusted to use its judgment. Where the library covers the topic, it prefers the library's treatment over its own knowledge. When the library *doesn't* cover the topic, it answers from general knowledge and says so plainly — without fabricating fake citations to make the answer look sourced. Best paired with frontier models that can be trusted to know the difference. Smaller local models can drift on this strategy; if you use it locally, watch for unsourced claims.

You can change the strategy in **Model Settings** → **Prompt strategy** at any time. Switching strategies does not change which sources are retrieved; it only changes how the model is told to use them.

Advanced

Model Profile Properties

The right-hand pane of **Model Settings** shows the properties of the selected profile. Most defaults are good, but a few are worth understanding.

Profile name. Whatever you'd like — this is just the label that shows in the toolbar's model picker.

Provider type. *Local (llama.cpp)* runs a `.gguf` model on your machine. *Claude (Anthropic)* and *OpenAI-compatible API* call a cloud (or local-server) endpoint over HTTP.

Model file *(local only)*. The full path to a `.gguf` file. Use the **...** button to browse.

Force CPU inference *(local only)*. Disables GPU offload even if a supported GPU is present. Leave this off unless you're troubleshooting GPU-related crashes; CPU inference is much slower.

API key, Model name, Base URL *(API only)*. Credentials and endpoint for the cloud provider. The base URL is pre-filled for Claude and OpenAI, and can be pointed at any OpenAI-compatible server (Ollama, LM Studio, a private deployment).

Context size. How much text — in tokens, roughly three-quarters of a word each — the model can hold in mind for a single turn. This includes the system prompt, the source passages the Librarian retrieved, the conversation so far, and the model's reply.

> **The tradeoff:** A larger context lets the Librarian feed the model more source material, more conversation history, and produce longer answers. That generally makes the answers better — more sources cited, more nuance, better follow-up handling. The cost is speed (large contexts are slower to process), memory (local models use more VRAM/RAM at higher contexts), and, for cloud models, money (you typically pay per token of input as well as output, so a question that includes 30,000 tokens of retrieved sources costs roughly six times as much as one that includes 5,000). **Auto** picks a sensible default for your hardware on local models, or 65,536 for cloud models. If a cloud model's bill is climbing faster than you'd like, drop this to 16,384 or 8,192 — answers will be a little tighter and a little less thorough, but each call costs a fraction as much.

KV cache type *(local only)*. The data type the model uses for its key/value cache during generation. *F16* is the default and most accurate. *Q8_0* roughly halves the cache's memory footprint with negligible quality loss — useful for fitting a larger context on the same GPU. *Q4_0* halves it again but starts to degrade quality on some models. Try Q8_0 first if you need more headroom.

Prompt strategy. Grounded, Balanced, or Conversational — see the *Prompt Strategies* section above.

Max source chunks. The maximum number of library passages fed to the model as evidence for a single question. **0 (the recommended default)** lets the Librarian fill as much of the available context as the source budget allows.

> **The tradeoff:** More chunks means more sources cited, broader coverage of a topic, and a higher chance that the most relevant passage is in the mix — but also slower responses, and (for cloud models) a bigger bill per question. Lower numbers (5–10) cost less and are faster, but the answer may miss relevant material that didn't make the cut, and comparisons between authors get thinner because there's less from each side to compare. If you're paying per token and find your typical question doesn't really need 25 sources, capping this at 10 or 15 is a reasonable cost-saver. If you want the most thorough answers possible, leave it at 0.

Save changes. Profile edits are committed when you click **OK**. **Cancel** discards them. Switching to a different profile in the list also commits the current edits.

Where the Librarian Saves Your Data

The Librarian uses three folders. The easiest way to open any of them is to copy the path below, press **Win+R** (or click the address bar in any File Explorer window), paste the path, and press Enter.

What	Location
Settings, model profiles, window placement	`%LOCALAPPDATA%\SwordSearcher\Librarian\`
Saved conversations (one JSON file per chat)	`%LOCALAPPDATA%\SwordSearcher\Librarian\history\`
Downloaded model files	`%PROGRAMDATA%\SwordSearcher\Librarian\models\`

`%LOCALAPPDATA%` and `%PROGRAMDATA%` are Windows shortcuts that File Explorer expands automatically. On a typical install they resolve to `C:\Users\<your name>\AppData\Local\` and `C:\ProgramData\` respectively. (`AppData` is hidden by default in Windows; using the shortcut bypasses that.)

The model location is a default — you can change it in the Download Model dialog (the **Download to:** field) if you'd rather keep models on a different drive. The new location is remembered for future downloads.

To back up your settings and chat history, copy the entire `%LOCALAPPDATA%\SwordSearcher\Librarian\` folder. To start fresh, close the Librarian and delete it.

Using a Model You Already Have

If you already have GGUF model files on your computer — for example, from **LM Studio** — you don't need to download them again. Point the Librarian at the existing file:

1. Open **Model Settings** (gear icon).
2. Click **Add** to create a new profile.
3. Set **Provider type** to **Local (llama.cpp)**.
4. Click the **...** button next to **Model file** and browse to your existing `.gguf` file.
5. Give the profile a name and click **OK**.

LM Studio normally stores its models under `%USERPROFILE%\.lmstudio\models\` (or `%USERPROFILE%\.cache\lm-studio\models\` on older versions), organized by publisher and model. Any `.gguf` file in there works.

The Librarian does not move or copy the file — it just remembers where it is. You can keep using the same file with both LM Studio and the Librarian.

Note on Ollama: Ollama stores its models in a packed format rather than as plain `.gguf` files, so the Browse dialog cannot use them directly. To use Ollama, set the provider type to **OpenAI-compatible API** instead, point it at your local Ollama server (typically `http://localhost:11434/v1`), and enter the Ollama model name (e.g. `gemma3:12b`). No API key is needed for a local Ollama instance.

System Requirements in Detail

The brief overview at the top of this help is enough for most people. This section is for anyone who wants to know exactly what hardware fits which model tier — useful when shopping for a new machine, deciding which model to download, or troubleshooting performance.

All tiers below assume **SwordSearcher Deluxe is already installed**. The Librarian shares SwordSearcher's module files, embedding model, and Bible semantic indexes. The library search indexes — the ones that make every commentary, dictionary, and book in the Deluxe library searchable by meaning — are built ahead of time and ship with the Librarian itself, so they're already on disk after install. Nothing is duplicated across the two installs.

Operating system, every tier: Windows 11 (64-bit) on an Intel or AMD x86/x64 processor. Earlier versions of Windows are not supported. **The four local-inference tiers below do not apply to ARM-based Windows PCs (Snapdragon X / Copilot+ PCs)** — local models cannot run on those machines. ARM users can still use Search Only mode or a Claude / OpenAI cloud profile; see the note in *Requirements* above.

Minimum (CPU only, smallest model)

The least hardware that produces usable answers. Runs Gemma 4 E4B at the Q3_K_M quantization on the CPU.

Component	Requirement
Processor	Modern 64-bit x86 CPU (Intel or AMD) with AVX2, **6+ physical cores**, ~3 GHz. Roughly 2018 or later for desktops; recent mainstream-tier laptops.
Memory	**16 GB RAM**
Graphics	Any. The GPU is not used at this tier.
Free disk space	6 GB (Librarian app + 4.6 GB Gemma 4 E4B Q3_K_M model + working headroom)
Internet	Required on first launch to download the model (~4.6 GB). Optional afterwards for local-only use.

Component	Requirement

What to expect: **3-8 words per second** of generation on a modern multi-core CPU (Ryzen 5000/7000, Intel 12th gen or newer); **2-4 words per second** on older AVX2-only systems. CPU inference is bottlenecked by memory bandwidth, so DDR5 is roughly 20-25% faster than DDR4 for this workload. Because the Librarian sends a large prompt to the model on every question (the retrieved sources plus conversation history), expect **15-60 seconds before the first word appears** depending on the CPU and the size of the conversation. Generation also slows as a conversation grows longer, since the model has more working memory to scan on every word. Fine for occasional questions; less comfortable for rapid back-and-forth.

Below the minimum tier: older 4-core ultra-low-voltage laptop CPUs (e.g. the 2019-era Surface Laptop with an i5-8250U, or any 8 GB ultrabook with integrated graphics) fall below this floor. Local models technically run, but the planning stage alone can take many minutes — long enough that the app appears frozen. The Librarian detects this configuration on startup and shows a slow-hardware warning on the welcome page. **Use Search Only mode or a Claude / OpenAI cloud profile on machines like that** — both work fine on hardware that local AI does not.

Recommended (consumer GPU)

A balanced setup that runs Gemma 4 E4B at Q6_K on a midrange GPU — much faster than CPU and high enough quantization that quality is excellent.

Component	Requirement
Processor	Modern 64-bit x86 CPU with AVX2, 6+ cores
Memory	16 GB RAM
Graphics	Discrete GPU with **8 GB VRAM**. NVIDIA (CUDA 12.4 or newer) or AMD/Intel (Vulkan 1.2 or newer).
Free disk space	8 GB (Librarian + 5.9 GB model + headroom)
Internet	Required on first launch

What to expect: **roughly 40-60 words per second** of generation once it begins. **The first word streams in 2-5 seconds** for short follow-ups, longer when the Librarian has packed the prompt with many source passages. Most questions complete in 12-30 seconds end-to-end. This is the tier most users should aim for — it is where the Librarian starts to feel conversational.

Best Performance (NVIDIA, 16 GB+ VRAM)

The highest local-inference tier. Runs Gemma 4 E4B at Q8_0 (near-lossless, the primary recommendation) with a generous working context, or the Gemma 4 26B-A4B mixture-of-experts model for stronger reasoning at comparable speed.

Component	Requirement
Processor	Modern 64-bit x86 CPU with AVX2, 8+ cores
Memory	32 GB RAM
Graphics	NVIDIA GPU with **16 GB or more VRAM** and CUDA 12.4 or newer. Examples: GeForce RTX 4080, 4090, 5080, 5090.
Free disk space	20 GB (E4B Q8_0 ~7.5 GB; 26B-A4B IQ4_XS ~14.2 GB if you keep both)
Internet	Required on first launch

What to expect: **roughly 50-75 words per second** of generation on E4B Q8_0 once it begins. **The first word streams in 1-3 seconds for short follow-ups, but on a freshly-packed query with the full 50,000-80,000-token source context, expect 5-10 seconds before the first word appears** — the model has to read all the retrieved passages first. Most questions complete in 10-20 seconds end-to-end, longer for very wide library retrievals. The working context can reach ~80,000 tokens on a 16 GB card, enough to feed a wide cross-section of your library into the model on every question. The 26B-A4B MoE is a noticeable quality leap and runs at similar speed because only a fraction of its parameters activate per word.

Cloud-only (no local model)

If you only ever use a Claude or OpenAI-compatible profile, the heavy lifting happens on the provider's servers and the local hardware requirements collapse to almost nothing.

Component	Requirement
Processor	Any modern 64-bit x86 CPU
Memory	4 GB RAM
Graphics	Any
Free disk space	50 MB (Librarian app + DLLs only — no model download)
Internet	Steady broadband connection during use; valid API key for the cloud provider

Even in cloud mode, the Librarian still does its own retrieval on your machine — it embeds your question with a local model, searches the on-disk indexes (the Bible indexes from SwordSearcher and the library indexes shipped with the Librarian), and assembles the source passages locally. Only the answer-generation step is offloaded to the cloud provider, and only the assembled prompt is sent over the network.

Notes

- **AVX2 is required.** The bundled `llama.cpp` build does not run on pre-2013 CPUs without AVX2.
- **No local models on ARM.** The Librarian does not run local AI models on ARM-based Windows machines (Snapdragon X / Copilot+ PCs and older Snapdragon laptops). On detection of an ARM/Adreno graphics adapter, the app refuses to load a local model and surfaces a notice on the welcome page directing the user to Search Only or a Claude / OpenAI cloud profile.
- **Slow-hardware warning at startup.** If the machine has no usable discrete GPU and falls below 6 physical CPU cores or 16 GB of RAM, the Librarian shows a warning on the welcome page recommending Search Only or a cloud profile. The warning is informational, not a block — local models will still attempt to load if a profile is configured, just very slowly.

- **GPU detection is automatic.** At launch the Librarian picks CUDA on NVIDIA cards and Vulkan on AMD/Intel GPUs. The only manual override is the **Force CPU inference** checkbox in Model Settings.
- **VRAM headroom matters.** A 7.5 GB model file does not fit comfortably on an 8 GB GPU because `llama.cpp` reserves about 1.25 GB for its compute buffer plus working memory for the KV cache. The download dialog shows a "fits" / "does not fit" indicator next to each tier so you do not have to guess.
- **No internet required after setup.** Once a local model is downloaded, the Librarian works fully offline. Cloud profiles always require internet.
- **Pre-downloaded models work.** If you already have GGUF files (for example, from LM Studio), point the Librarian at them via *Model Settings → Add → Local → Browse...*. There is no need to re-download.

Glossary

A short reference for terms used in this help file and in the Librarian's settings.

AI / Artificial Intelligence. Used here as an informal label for the language model that writes the Librarian's answers. The "intelligence" is statistical pattern-matching, not understanding — useful, but not a substitute for your own judgment.

API / API key. An *API* (Application Programming Interface) is a way for programs to talk to a service over the internet. An *API key* is a password-like string that proves you're authorized to use a particular service (Claude, OpenAI). API keys are tied to billing — keep yours private.

Cloud model. A language model that runs on a provider's servers (Anthropic's Claude, OpenAI's GPT, etc.). Your question is sent over the internet to the provider, processed there, and the answer is sent back. Generally higher quality than local models but costs money per question and exposes your queries to the provider.

Chunk. A short passage of text — typically a paragraph or two — carved out of a commentary, dictionary, or book entry when the library was indexed. Semantic search retrieves chunks rather than whole entries, so the model sees just the

relevant slice of a long article instead of the entire thing. Each chunk remembers which module and entry it came from, which is how citations link back to the right place in SwordSearcher. The number of chunks fed to the model per question depends on two settings that work together: **Context size** sets the overall budget (system prompt + chunks + conversation history + reply all have to fit), and **Max source chunks** sets an upper limit on how many chunks are allowed in that budget. Whichever is tighter wins — a large context with Max set to 10 still tops out at 10, and a huge Max setting won't help if the context is small. Leaving Max at **0** (the default) lets the budget alone decide.

Context / Context window. The amount of text the model can hold in working memory for a single turn — measured in tokens. The context window has to fit the system instructions, the source passages the Librarian retrieved, the conversation so far, and the model's reply. Larger context windows allow more thorough answers but are slower and (for cloud models) more expensive.

GGUF. The file format used by `llama.cpp` for local language models. A `.gguf` file contains the model's weights and metadata in a single self-contained file, typically several gigabytes. Models you download in the Librarian arrive as `.gguf` files.

GPU / Graphics Processing Unit. The chip that drives your computer's display. Modern GPUs are very good at the math language models need, so running a model on the GPU is usually 5–20× faster than running it on the CPU. *VRAM* (Video RAM) is the GPU's dedicated memory — the more VRAM you have, the larger a model you can fit on the GPU.

Hallucination. When a language model invents something that sounds plausible but isn't true — a fake quote, a misattributed source, a verse reference that doesn't exist. The Librarian's source-grounded design and citation linking are intended to make hallucinations easy to catch: if a citation doesn't open the entry it claims, that's a hallucination. Always verify important claims against the cited source.

Inference. The act of running a language model to generate a response. "Local inference" means running on your computer; "cloud inference" means running on a provider's servers.

KV cache. A working buffer the model uses while generating a response. Larger contexts need a larger KV cache, which uses more memory. The *KV cache type* (F16, Q8_0, Q4_0) controls how much memory the cache consumes — see *Model Profile Properties* above.

llama.cpp. The open-source software the Librarian uses to run local models. It supports a wide range of models in `.gguf` format and runs efficiently on CPUs and most GPUs.

LLM / Large Language Model. The kind of AI model that powers the Librarian's answers. Trained on huge amounts of text, LLMs predict what word should come next given everything that came before. That simple mechanism, applied at scale, is what produces fluent answers.

Local model. A language model that runs entirely on your own computer. Slower and (at consumer hardware tiers) lower-quality than cloud models, but completely private — no question or library content is ever sent over the internet.

Prompt. The full set of instructions and context handed to the model before it generates an answer. The Librarian assembles a prompt for each question, combining a system prompt (the strategy instructions), the retrieved sources, your conversation history, and your current question.

Prompt strategy. Three preset variations of the system prompt — Grounded, Balanced, Conversational — that control how strictly the Librarian sticks to the cited sources versus how freely it uses general knowledge to frame an answer. See *Prompt Strategies* above.

Quantization. A technique for shrinking a model file by storing its numbers at lower precision. The labels you see in the download list (Q3_K_M, Q4_K_M, Q5_K_M, Q6_K, Q8_0) describe the quantization level — the number is roughly the bits per weight. Lower numbers (Q3) produce smaller files and run faster but degrade answer quality. Higher numbers (Q8_0) are nearly indistinguishable from the original full-size model but take more memory. Q5_K_M and Q6_K are common sweet spots.

RAG / Retrieval-Augmented Generation. The technique the Librarian is built on. Rather than letting the model answer from memory alone, the Librarian first

retrieves relevant passages from your library and feeds them to the model as evidence. The model's job is to synthesize an answer from that evidence and cite its sources. This is what makes the Librarian's answers source-grounded and verifiable.

RAM / Random Access Memory. Your computer's main working memory. Local models load into RAM (or VRAM, if running on the GPU). The Librarian's recommended hardware tiers are based on how much RAM/VRAM is needed to fit a model of a given size at a given quantization.

Semantic search. Searching by *meaning* rather than by matching words. The Librarian converts your question into a numerical "fingerprint" and finds passages whose fingerprints are similar — so a search for *"sin nature"* can surface a commentary that talks about *"depravity"* or *"the carnal mind"* without using the literal words you typed. This is how the Librarian retrieves source material to feed the model.

Streaming. Showing the model's answer word-by-word as it's generated, instead of waiting for the full reply. The Librarian streams every answer so you can start reading immediately and click **Stop** if it's heading the wrong direction.

System prompt. The instructions given to the model at the start of every conversation, telling it what it is, what to do, and how to format its output. The Librarian's system prompts are built into the prompt strategies and are not user-editable.

Token. The unit a language model reads and writes — roughly three-quarters of an English word. "Tokens" are the unit context windows are measured in: a 32,768-token context window holds about 25,000 words of input plus output combined. Cloud providers also bill per token.

VRAM / Video RAM. The GPU's dedicated memory. Running a local model on the GPU requires the model file (and its KV cache) to fit in VRAM. See *GPU* above.