ChatGPT

llms.txt and llms-full.txt: Purpose, Function, and Usage in AI Contexts

Overview of llms.txt and llms-full.txt

llms.txt is a newly proposed standard for making website content more accessible and understandable to AI **Large Language Model (LLM)** agents. It serves as a **machine-readable entry point** for LLMs, analogous to how a `robots.txt` file guides web crawlers [1]. In essence, an `llms.txt` file is a **curated index** of a website's most important content for AI, typically provided in Markdown format. It might include a brief overview of the site or project (title and description) and a **list of key pages or resources with short descriptions**, structured for easy parsing [2] [3]. The goal is to help an AI quickly identify "high-value" pages (such as documentation, FAQs, product info, etc.) without having to scrape the entire site blindly [4] [5]. In practice, `llms.txt` acts similar to a sitemap or an outline **optimized for inference** (answer-generation) rather than search indexing [1]. For example, an `llms.txt` might look like a Markdown document with a project title, a one-paragraph summary, and a list of important links (each link accompanied by a one-line description) [6]. This structured approach allows an LLM to **grasp the site's content hierarchy** at a glance and retrieve relevant information more directly [7].

llms-full.txt is a companion convention that goes a step further. While `llms.txt` points to important pages, `llms-full.txt` **contains the full content of those pages (and often the entire documentation or knowledge base) in one file** [4] [8]. In other words, `llms-full.txt` collapses a collection of documents into a single large Markdown file – effectively a one-stop **knowledge dump** for the AI. This file is not officially part of the llms.txt *standard* proposal, but many teams have adopted it informally [9]. The rationale is that an AI agent could retrieve **all the detailed content in one request**, without needing to follow multiple links or fetch numerous pages [10]. For example, a documentation site might expose `/llms-full.txt` at its root, which concatenates every page of the docs (API references, guides, tutorials, etc.) in sequence [8]. The open-source project *aws-cloudformation-media-downloader* demonstrates this pattern: it added a `docs/llms.txt` as a high-level map of the documentation, and a `docs/llms-full.txt` as a *"concatenated knowledge base of all documentation"* [11]. Essentially, `llms-full.txt` provides the **raw textual content** that an LLM might need for deep answers, whereas `llms.txt` provides a **structured summary and pointers**.

How llms.txt and llms-full.txt Improve LLM Agent Efficiency

These files are designed to make AI agents more **efficient** in finding and using information. Normally, an LLM-based agent trying to learn about a project or site would have to crawl HTML pages (which include navigation menus, ads, and other clutter) and possibly traverse many links. An `llms.txt` greatly streamlines this by presenting a clean, distilled index in plain text (Markdown). This reduces the overhead of parsing irrelevant HTML boilerplate and focuses the AI on the **essential content structure**. In fact, some documentation platforms now automatically serve Markdown to known AI user-agents instead of HTML, cutting down token consumption by *over 90%* [12]. By pointing the AI to the most relevant pages and giving

one-line summaries, `llms.txt` helps the model decide *what to read first* for a given query, improving both **accuracy and speed** of AI responses [7]. As one explainer puts it, *"the point of LLMs.txt is to help an LLM find the best information about your website in the most direct way"* [3]. In effect, it's similar to how a sitemap aids search engines, except here the benefit is for inference-time retrieval: the LLM doesn't waste time or context window space on less relevant pages [5].

The `llms-full.txt` further boosts efficiency in scenarios where the AI needs comprehensive understanding. Rather than fetching many pages individually (which might require multiple web requests or prompt iterations), an agent can fetch a single `llms-full.txt` and obtain **all the content in one go** [10]. This is especially helpful for tools that implement Retrieval-Augmented Generation (RAG) – they can ingest the full text and index it internally. For instance, in an IDE or chatbot setting, loading `llms-full.txt` into the context means the AI has the entire documentation at its fingertips, ready to answer detailed questions without additional web lookups [13]. This was one motivation in the *aws-cloudformation-media-downloader* project's AI optimization: by providing a pre-packaged full context (`llms-full.txt` and also a packed code context via Repomix), the maintainers enabled LLM agents (like Google's Gemini or Anthropic's Claude) to consume the project knowledge base **quickly and with minimal fuss** [14]. Moreover, delivering content as plain text in a single file can mitigate the issue of small context windows – rather than the model trying to ingest an entire site structure with multiple fetches, the curated content can be summarized or chunked as needed. Jeremy Howard (who proposed the spec) notes that having *"a single place where all of the key information is collated"* is useful not for training, but for *helping LLMs serve users* with up-to-date info [15]. In summary, **llms.txt/llms-full.txt improve efficiency by reducing unnecessary crawling, focusing the context on important data, and minimizing tokens wasted on irrelevant text** [12] [7].

It's worth noting a trade-off: `llms-full.txt` files can become **very large** (hundreds of thousands of tokens for big documentation sets) [16]. While this consolidates information, it might exceed an LLM's context window. Therefore, some agents will prefer using `llms.txt` to identify relevant sections and then fetch those separately, rather than loading an enormous context indiscriminately [10]. In practice, an efficient strategy is often to use `llms.txt` as the starting point (for high-level navigation) and only resort to `llms-full.txt` or individual page fetches when deeper detail is required [10]. Nonetheless, the presence of these files gives AI agents flexible options: quick overview versus full content, whichever fits their context budget.

Comparison with robots.txt, humans.txt, and Other Standards

The concept of `llms.txt` is inspired by earlier *plain-text conventions* used on websites. In function, it's often compared to **robots.txt**, which is a file that web crawlers (like Googlebot or Bingbot) read for instructions on which pages to crawl or avoid. Like robots.txt, an `llms.txt` file is placed at a standard location (usually `https://yourdomain.com/llms.txt`) and is meant for **machine consumption** [17]. However, the similarity largely ends there. Robots.txt is about **access control and indexing rules** (it tells search engines what to *not* crawl or where to find sitemaps), whereas `llms.txt` is about **content discovery and summarization** for AI (guiding LLMs to what *to read* and understand) [18] [19]. In fact, `llms.txt` can complement robots.txt: the former provides context and pointers to *allowed* content, while the latter could still be used to disallow certain paths if needed [19]. Importantly, at present `llms.txt` is a voluntary convention – not a formally recognized standard by all AI providers – so compliance by bots is not guaranteed (similar to how obeying robots.txt is voluntary, though widely honored by search engines) [20].

Another related file is **humans.txt**, which is a textual note some sites include to credit the developers or provide Easter eggs for curious human visitors. As the name suggests, `humans.txt` is *not for bots at all* (search engines or AI), but rather a fun human-readable blurb. In contrast, `llms.txt` is explicitly **meant for AI "crawlers" or agents** to read [17]. One source puts it succinctly: *"Humans.txt is more informational and not read by bots. Contrarily, llms.txt is meant only for AI crawlers."* [21]. So while all these files share a format of plain text with potential guidelines or info, their audiences differ: robots.txt -> search bots; humans.txt -> people; **llms.txt -> AI models** (LLMs) and tools using them. We can also compare `llms.txt` to a **sitemap or documentation index**: a sitemap (often XML) lists all pages for search indexers, whereas `llms.txt` offers a **curated overview** of content specifically tuned for language model consumption [19]. It's not about listing *every* URL, but the most useful ones, possibly with explanation. In fact, the llmstxt.org specification emphasizes that Markdown is used (instead of XML) to be easily readable by both humans and LLMs, yet still structured in a predictable way for parsing [22] [23].

To illustrate the analogy: If `robots.txt` is a **gatekeeper** telling web robots where they can't go (and occasionally pointing them to a sitemap of where they can), then `llms.txt` is a **welcome guide** for AI, pointing out the highlights of the site and where to find important knowledge [1] [3]. And if `humans.txt` is a little thank-you note on the doorstep for human visitors, `llms.txt` is a **cheat-sheet** handed to the AI visitor so it can serve users better. It's also worth noting that there are other similar `.txt` files like `ads.txt` (for advertising networks) or `security.txt` (to tell security researchers how to report issues) [24]. Each serves a niche audience, and `llms.txt` is simply the latest, aimed at the burgeoning audience of AI agents crawling the web for information [25].

Current support: As of late 2025, `llms.txt` is still an *emerging* practice. Major search/AI companies have mixed stances – for example, Google's representatives have indicated their AI search (SGE) currently does *not* use `llms.txt` and relies on normal SEO and content quality [26]. OpenAI's GPTBot, on the other hand, has been observed actively crawling `llms.txt` files on sites (reportedly hitting some sites' llms.txt every 15 minutes) [26], suggesting that at least some LLM developers are experimenting with it. So while not a universally adopted standard yet, the convention is gaining traction in communities preparing for "AI-first" content discovery (sometimes dubbed **Generative Engine Optimization (GEO)**, akin to SEO [27]). In summary, `llms.txt` is conceptually akin to established files like robots.txt/humans.txt, but it occupies its own space: **guiding AI content consumption rather than controlling indexing or delivering human-focused info**.

Best Practices for Using llms.txt and llms-full.txt

If you choose to implement `llms.txt` / `llms-full.txt` for your site or project, consider these best practices to ensure they are effective and correctly used by LLM agents:

- **Follow the Standard Format:** The community-driven spec (see **llmstxt.org**) recommends that `llms.txt` be a Markdown file with a specific structure [28]. Start with an H1 heading that is the site or project name (this is the only required element) [29]. Following that, it's good to include a brief description or summary of the project, perhaps as a paragraph or blockquote, highlighting key points or context [2]. After the overview, list important sections or pages. Use secondary headings (H2) to organize these lists if needed (for example, "## Documentation" or "## APIs"), then under each, provide a bullet list of links. Each list item should be a Markdown hyperlink to a relevant page, followed by a colon and a short note about what that page contains [2] [30]. This way, an AI scanning

the file knows *"Link title – description"* for each important resource. Avoid using too deep a heading hierarchy; keep it simple and scannable. A mock example from the spec is:

```
# MyProject
> Open-source toolkit for XYZ (One-line project description)

## Getting Started
- [Installation Guide](https://example.com/docs/install): Steps to install and
configure.
- [Quickstart Tutorial](https://example.com/docs/quickstart): Introductory usage
guide.

## Reference
- [API Documentation](https://example.com/docs/api): Full API reference.
- [CLI Commands](https://example.com/docs/cli): Command-line tool usage.

## Optional
- [Changelog](https://example.com/docs/changelog): Version history (less
critical).
```

This illustrates a clear structure: top-level title, a short summary, and sections of key links with descriptions. (Notably, a section titled "Optional" can be used for less crucial links; by convention, an AI may skip that if it's trying to trim context [31].) Sticking to this format will make it easier for any tooling or agent expecting `llms.txt` to parse it reliably.

- **Place the Files in Standard Locations:** Typically, `llms.txt` should live at the **root of your website** (e.g. `https://yourdomain.com/llms.txt`) or the root of your documentation site. Many implementations also provide it at sub-paths (for example, if documentation is under `/docs/`, you might have `/docs/llms.txt` as well) [32]. The open-source `aws-cloudformation-media-downloader` project, for instance, put these files in its `docs/` directory (likely to be served on a documentation site or GitHub Pages) [33]. Ensure that the file is publicly accessible (no authentication required) so that an AI agent can fetch it. For `llms-full.txt`, place it at the root of the site or docs hierarchy (commonly at the root, since it's one per site) [34]. The `llms-full.txt` should ideally enumerate or include *all relevant content pages*. In documentation use-cases, this means the entirety of the docs in one file [8]. In other contexts, it could even include important knowledge base articles or even the README and Wiki of a project. Essentially, if `llms.txt` lists "high-value" pages, `llms-full.txt` should contain the **full text of those high-value pages concatenated** (usually in a logical order or with clear separators). Always include at least a top-level title in `llms-full.txt` as well (many auto-generators just repeat the site title at top).

- **Be Concise and Curate**: *Curation* is key for `llms.txt`. Don't list every single page of your site – that's what a sitemap does. Instead, list what an AI **should focus on**. For example, a software project might include links to its installation guide, main usage docs, API reference, and troubleshooting FAQ, while omitting less useful pages like press releases or auto-generated legal policies. Each link's description should be one sentence (or even a clause) that captures the essence. This helps the LLM

decide relevance quickly [35] [3]. In `llms-full.txt`, you have more leeway to include content verbatim, but you might still exclude truly irrelevant text (like navigation menus, or repetitive footer info) when compiling it. Some tooling automates this extraction to include only the main content of pages. The idea is to maximize the signal-to-noise ratio for the AI.

- **Keep Content Up-to-Date:** Treat these files as living documents that should be updated when your content changes. If you add a major new documentation page or a crucial blog post that you want AI to know about, update `llms.txt` to include it. If your documentation text changes, regenerate or edit `llms-full.txt` to stay current. Stale or broken links in `llms.txt` defeat the purpose, and outdated information in `llms-full.txt` could mislead AI answers. If your site has a build process or CI, consider automating the generation of `llms.txt` (and `llms-full.txt` if feasible) to avoid manual errors. For instance, some platforms like **Mintlify** and **Fern** auto-generate these files from the existing docs structure [36] [37]. If doing it manually, a simple script could concatenate Markdown files for `llms-full.txt`. Always verify the file renders or is structured correctly (you can open the `.txt` in a browser or Markdown viewer to check formatting).

- **Use AI-Specific Tags if Available:** A few documentation platforms support special tags to fine-tune content for AI. For example, Fern allows marking certain parts of docs with `<llms-only>` (included in AI-facing outputs like llms.txt but hidden from human-facing docs) or `<llms-ignore>` (visible to humans but omitted from AI outputs) [38] [39]. If you have access to such features, you can include extra explanatory text for AI (like deeper technical context that a normal user might not need) inside an `llms-only` block – this will enrich what the AI sees in `llms.txt` / `llms-full.txt` without cluttering the regular docs [40]. Conversely, you might want to exclude content that is irrelevant or potentially confusing for the AI (marketing fluff, for example) using an ignore tag [39]. Even if your platform doesn't support these exact tags, you can manually decide to put additional notes in the llms files that aren't present on the website, or omit sections that don't help answer questions. The key is to tailor the AI-facing content to be **maximally helpful for answering user queries**.

- **Test with an LLM Agent:** After setting up `llms.txt` (and full file), it's wise to simulate how an LLM would use them. You can use tools like OpenAI's GPT-4 browsing (if available), Bing Chat, or smaller-scale scripts to fetch your llms.txt and see if the content is actually helpful. For instance, ask an AI (with browsing enabled) a question that should be answerable from your documentation, and see if it picks up info from your `llms.txt` / `llms-full.txt`. If the answers still seem off-target, you might need to adjust the file's contents or structure. Additionally, monitor your server logs if possible – you might notice hits to `/llms.txt` from various bots. Over time, analytics could show which AI providers or tools are fetching it [41]. This feedback loop can guide you in improving the files.

- **Mind the Context Window:** If you maintain an `llms-full.txt`, consider the size. As noted, a very large file can't be fully loaded into some models' prompt. As a best practice, you might keep extremely verbose sections (like entire API schemas) in a separate "optional" part or just referenced via link. Some sites implement query parameters to let the AI or user agent request a slimmed version – for example, filtering out code samples or picking a specific programming language subset [42]. While these advanced features are not standard, they highlight the idea of keeping the content as concise as possible for AI consumption. A good practice is to ensure your `llms-full.txt` is **plain text only** (no binary content, obviously), and maybe compressible (if a client supports fetching a compressed version). But don't worry too much about file size unless it's truly enormous; retrieval tools with RAG can handle chunking large docs [43].

Finally, remember that any information in `llms.txt` or `llms-full.txt` is **public** (just like anything on a public website). Do not include API keys, private internal details, or anything you wouldn't want to be widely known. The files should guide AI to *public* knowledge about your project. Treat them as an extension of your documentation or homepage – they should reflect the same messaging and transparency.

Leveraging llms.txt in Open Source Projects and AI Interactions

Open source projects stand to gain significantly by leveraging `llms.txt` and `llms-full.txt` to guide AI interactions. As AI assistants become more common for helping developers (e.g. GitHub Copilot, Cursor, or GPT-based CLI helpers), maintainers can **pre-configure how an AI perceives their project**. A prime example is the *aws-cloudformation-media-downloader* repository's recent update (the "AI Optimization Suite") which explicitly added these files as **"Agent Entry Points."** In that project, `docs/llms.txt` provides a high-level map of the repository's documentation, and `docs/llms-full.txt` provides the entire knowledge base of documentation in one place [11]. This means that any LLM agent (like an AI coding assistant) encountering the repository can immediately find a roadmap to the project's important information. The PR description notes that this helps agents like Google's Gemini and Anthropic's Claude better integrate with the repo [44] [11]. In essence, the maintainers are saying: *"Here's everything you need to know about our project, dear AI, neatly organized and ready for you to consume."*

Other open source and tech projects have begun doing similar things. Documentation platforms such as **Fern** and **Mintlify** automatically generate llms.txt files for all docs sites they host, because they recognize that making docs AI-friendly benefits both the users and the projects [37] [45]. For example, ElevenLabs and Cash App's developer docs are cited as having live `llms.txt` and `llms-full.txt` endpoints [46] [34]. Even the popular LangChain project provided `llms.txt` indexes for their Python and JS docs, and a combined `llms-full.txt` – enabling IDE plugins or other tools to easily load LangChain documentation into an assistant [47]. The maintainers of LangChain explicitly highlight how these files allow LLM agents to access programming documentation more effectively, especially in IDEs or chatbot scenarios [48]. They also caution about context size, reinforcing that developers should use `llms.txt` (index) for targeted navigation and `llms-full.txt` when they have the ability to chunk or handle large files [10].

What to include: In an open source project, consider including in `llms.txt` the key resources that an AI developer assistant or support agent should know. This often includes the README or introduction, contributor guides, API docs, usage examples, and any "getting started" tutorials. If your project has a `CONTRIBUTING.md` or `ARCHITECTURE.md`, those are great to list as well (since AI might get questions about how to contribute or the project's design). You can also include links to external resources: for instance, if there's a crucial blog post or a design document in a wiki, link it. The descriptions should mention what the file is ("API reference for module X", "Troubleshooting guide for common errors", etc.). For `llms-full.txt`, you might literally concatenate those markdown files (`README.md`, `docs/*.md`, etc.) into one big file. Ensure there's clear separation (perhaps a header before each concatenated section) so that when an AI reads it, it knows where one document ends and the next begins.

Open source maintainers can also use these files to **set boundaries or guidance** for AI. For example, you might include a note in `llms.txt` like: *"This project follows certain coding conventions (see Conventions doc) and any AI-generated code should adhere to them."* In fact, the example project included a `.gemini/instructions.md` alongside llms files to give system-level instructions to an AI (Gemini) about coding rules [11]. While those instructions are model-specific, it showcases the idea that you can proactively steer AI

behavior. In `llms.txt`, one could imagine adding a brief note in the summary or an extra section to highlight project ethos, licensing (e.g. "Note: This project is GPL-3.0 licensed – AI outputs should respect license terms."), or other important caveats for AI usage. Since `llms.txt` is both human- and machine-readable, this doubles as guidance to anyone building AI on top of the project.

Using `llms.txt` in open source projects also fosters **transparency and consistency**. Anyone (or any agent) who accesses the project via an AI interface will be drawing from the same canon of information – the one provided by the maintainers. This reduces the risk of the AI missing context or hallucinating details because it overlooked a part of the docs. It's akin to providing an official FAQ or "cheat sheet" that the AI can rely on. As a best practice, open source projects can leverage templates or generators (perhaps as part of their docs build) to create these files. Projects might even collaborate on improving the convention; for instance, if many projects adopt llms.txt, AI tool developers could standardize how their bots look for and use these files. There are already community discussions about integrating `llms.txt` support into platforms and plugins [49] – for example, WordPress plugin developers have started adding support to generate llms.txt files for sites using popular SEO plugins [50].

In summary, open source projects can use `llms.txt` and `llms-full.txt` to **guide AI interactions by surfacing the most relevant information and providing complete context when needed**. This improves the chances that an AI (be it an answer bot, a coding assistant, or a search engine's AI snippet generator) will produce accurate and helpful outputs about the project. It's a forward-looking way to make your project a "first-class AI citizen," as the Media Downloader project PR aptly put it [44]. By curating what the AI sees, maintainers can indirectly shape the narrative and correctness of AI-generated answers about their project. And as AI integration in search and tooling grows, adopting conventions like `llms.txt` is a relatively low-effort step that could significantly boost your project's visibility and usability in the AI-driven future of information sharing.

Sources:

- Thenuka Karunaratne, *"What is LLMs.txt + LLMs-Full.txt? A proposed technical standard for AI visibility across the web."* (May 22, 2025) [1] [4] [9]
- Mintlify Documentation, *"llms.txt – Optimize your docs for LLMs to read and index."* [45] [7] [51]
- Fern (Build with Fern) Docs, *"llms.txt and llms-full.txt"* [12] [52] [8]
- Dataprovider.com Tech Blog, *"Optimizing websites for the future of search – LLMs.txt explained"* (Jan 15, 2025) [5] [3] [15]
- Digital4Design Blog, *"What is llms.txt? 2025 Guide for WordPress"* [21] [17] [20]
- **llmstxt.org** – *"The /llms.txt file"* (Jeremy Howard's proposal, Sep 3, 2024) [53] [28] [19]
- LangChain LangGraph Docs, *"Differences Between llms.txt and llms-full.txt"* (2025) [10] [16] [48]
- GitHub – aws-cloudformation-media-downloader, *Pull Request #202: "AI Optimization Suite (Context, Semantics, Compliance)"* (Dec 21, 2025) [11] [14] [44]

---

[1] [4] [9] What is LLMs.txt LLMs-Full.txt?
https://journal.withdaydream.com/p/what-is-llms-txt-llms-full-txt

[2] [19] [22] [23] [28] [29] [31] [53] The /llms.txt file – llms-txt
https://llmstxt.org/

3 5 15 24 25 Optimizing websites for the future of search? LLMs.txt explained
https://www.dataprovider.com/blog/tech/llms.txt-explained

6 7 30 35 36 45 51 llms.txt - Mintlify
https://www.mintlify.com/docs/ai/llmstxt

8 12 32 34 37 38 39 40 41 42 46 52 llms.txt and llms-full.txt | Fern Documentation
https://buildwithfern.com/learn/docs/ai-features/llms-txt

10 13 16 43 47 48 llms.txt
https://langchain-ai.github.io/langgraph/llms-txt-overview/

11 44 feat: AI Optimization Suite (Context, Semantics, Compliance) by j0nathan-ll0yd · Pull Request #202 · j0nathan-ll0yd/aws-cloudformation-media-downloader · GitHub
https://github.com/j0nathan-ll0yd/aws-cloudformation-media-downloader/pull/202

14 33 feat: AI Optimization Suite (Context, Semantics, Compliance) by j0nathan-ll0yd · Pull Request #202 · j0nathan-ll0yd/aws-cloudformation-media-downloader · GitHub
https://github.com/j0nathan-ll0yd/aws-cloudformation-media-downloader/pull/202/files

17 18 20 21 26 27 What Is llms.txt? Learn How to Add It to WordPress (2025)
https://www.digital4design.com/blog/llms-txt-wordpress-guide/

49 Add support for llms.txt (similar to robots.txt, humans.txt, ads ... - GitHub
https://github.com/nystudio107/craft-seomatic/issues/1641

50 Humans-txt Plugins — WordPress.com
https://wordpress.com/plugins/browse/humans-txt/