LEVERAGING LLMS FOR AUTOMATIC FORUM SCRAPER GENERATION

Miloš Pavković[1]*,
[0000-0001-7776-6045]
Jelica Protić[2],
[0000-0003-0846-0290]
Petar Kresoja[1]
[0009-0008-3343-1540]

[1]Singidunum University,
Belgrade, Serbia

[2]School of Electrical Engineering,
University of Belgrade,
Belgrade, Serbia

Abstract:

Web forums contain valuable user-generated content (UGC), but crawling them presents a challenging task due to the differences between forum technologies and structures. This paper proposes a general approach that uses Large Language Models (LLMs) to automatically detect the forum technology (e.g., phpBB, vBulletin, SMF, Discuz!) and generate a web scraper for that forum's layout, structure, and pagination. LLM first identifies the platform of a given forum by analysing its HTML patterns after it generates code to efficiently collect available posts and threads that are publicly available and don't require user registration.

Several state-of-the-art LLMs are evaluated (GPT-4, Claude 2, and Mistral 7B) for this task, comparing their speed, accuracy, and reliability in generating functional scraping code. A proof-of-concept functionality was demonstrated on a chosen phpBB forum technology by crawling its content with LLM-generated Python code.

Experimental results show that the LLM-generated scrapers can successfully retrieve forum posts with high accuracy, matching manually coded crawlers while adapting automatically to different forum structures. The findings suggest that LLMs can significantly improve forum data collection, avoiding manual per-site adjustments and reducing duplicate content in incremental crawls.

Keywords:

Large Language Models, Web Scraping Automation, Template Detection, Data Retrieval.

INTRODUCTION

Online discussion forums are a valuable source of user-generated content, containing discussions, Q&A, reviews, and community knowledge [1]. Retrieving this content has applications in domains like social media analysis, customer feedback evaluation, and knowledge extraction. However, forums present unique challenges for web crawlers: content is spread across pages, threads and posts, and each forum technology (phpBB, vBulletin, SMF, Discuz!, etc.) has its own HTML layout and navigation structure. New posts continually shift older posts to new pages, complicating incremental crawling and often leading to duplicate data retrieval if not handled properly. Traditional forum crawlers either rely on manually written code for each forum technology or attempt a one-size-fits-all strategy that may not capture details related to that specific website.

Correspondence:
Miloš Pavković

e-mail:
mpavkovic@singidunum.ac.rs

DOI: 10.15308/Sinteza-2025-78-85

SInFo [1], a recent structure-driven forum crawler, highlights these issues: it targets the latest content by leveraging forum-specific URL patterns and pagination routes while remaining generic across platforms. SInFo achieved an average of 92.6% new content per recrawl cycle, demonstrating the importance of understanding forum structure to avoid redundant downloads. Despite such advances, implementing a new scraper for each forum or generalizing across all forums still demands significant human effort in analysing HTML and building site specific parsing rules.

In this paper, a new LLM-driven approach for forum crawling is proposed. The key idea is to combine the LLMs understanding of text and code generation with the structured nature of forums. The system first detects the forum technology on a target website by examining distinctive technology features (such as footer text, URL signatures, and HTML layout structure). Once identified, prompts are sent to the LLM to generate a scraping script specific to that platform's structure (e.g., how threads, pages and posts are organized). This two-step approach ensures that details of each forum technology are properly captured and parsed. For example, if the forum technology is recognized as phpBB, the LLM can use knowledge of phpBB's page numbering system and thread HTML structure to produce an accurate crawler. If instead it's an SMF forum, the LLM would know to look for *index.php?topic=* patterns and the corresponding navigation scheme that are unique for this type of technology. This approach is evaluated using three different popular LLMs – GPT-4 by OpenAI [2], Claude 2 by Anthropic [3], and Mistral 7B [4] (a smaller open-source model) – to compare their performance in code generation for this task.

This work, (i) introduces a standardized approach for forum crawling that automatically adapts to different forum technology software using LLMs, (ii) presents a comparison of multiple popular LLMs (proprietary and open source) in generating web forum scraping code in terms of speed, accuracy, and reliability, and (iii) demonstrates through an experimental setup that an LLM-generated scraper can successfully crawl a real forum, matching the efficiency of manually written crawlers. The proposed approach can target only publicly viewable forums, focusing on open content – user logging is explicitly excluded so as bypassing the CAPTCHAs and other restrictions.

2. RELATED WORK

The recent study of focused web crawling advancements has been made by integrating semantic analysis and optimization algorithms to enhance performance. For instance, Liu et al. introduced a focused crawler that combines a semantic disambiguation graph with a semantic vector space model to improve the retrieval of topic-relevant web pages [5]. Similarly, Huang et al. proposed a crawler that constructs a semantic graph to eliminate ambiguous terms and employs a genetic algorithm to optimize weighting factors, resulting in improved acquisition rates and relevance [6]. While these approaches demonstrate the potential of semantic understanding and intelligent learning in focused crawling, they may not fully address the challenges of dynamically detecting and adapting to various forum structures and technologies without extensive manual configuration.

In early research on web forum crawling the challenges were complex navigation and duplicate content. FoCUS (Forum Crawler Under Supervision), which used machine learning, identified forum-specific URL patterns and page templates [7] to handle complex navigation. FoCUS learned regular expressions for forum thread URLs and navigational links by training on annotated forums, enabling it to crawl forums at scale with minimal updates and rules. However, it required supervised training and did not generalize to unseen forum layouts without additional examples [7]. On the other hand, structure-driven crawling methods used the predictable layout of forums: for example, the work of Pavković and Protić on SInFo (Structure-Driven Incremental Forum crawler) proposed a generic two-phase strategy [1]. In SInFo, the crawler first separates index pages (listing threads) from content pages (showing posts), then uses the forum's navigational structure to find the page containing the newest content. By observing the URL format for each forum technology, SInFo could target directly the latest posts, avoiding re-fetching pages seen in previous crawls.

The LLMs ability to understand language, text and code has started the interest in using them to automate web scraping tasks. The core advantage of LLMs is their ability to understand context and generalize on it, which helps in interpreting various webpage structures. LLM-powered scrapers can interpret and understand complex website structures, making them more effective than traditional scraping tools [8]. Unlike traditional scrapers that can easily break when the site's layout changes since they rely on delicate CSS selectors or XPaths, LLM-based scrapers leverage natural language understanding to adapt to changes.

Several practical systems have been developed that integrate LLMs into the scraping pipeline. ScrapeGhost by Turk is an experimental library where the user provides a target URL and a desired output schema; the library then prompts GPT-4 to extract the target data [9]. It removes the need to manually write parsing code for each site. Similarly, FireCrawl [10] is an open-source tool that crawls a website and returns content in a clean, structured format (like Markdown or JSON), suitable for feeding into LLM applications. In a more general sense, frameworks like LangChain [11] started including web browsing and scraping capabilities through LLMs, where the model can be instructed to use a browser tool to navigate pages and then parse them with its internal reasoning [12]. These agentic approaches like AutoGPT or BabyAGI can crawl web pages by iteratively deciding which link to follow next and when to stop, using the LLM's output as the controller. While flexible, they often experience high token usage and latency, and need careful prompting to stay focused to avoid the agent getting lost on irrelevant links.

3. METHODOLOGY

The approach proposed in this work consists of two main stages: (A) Forum Technology Detection and (B) Scraper Code Generation, followed by an execution and evaluation phase. In the following chapters, each stage will be described in detail, including how LLMs are utilized and tuned for the task.

3.1. FORUM TECHNOLOGY DETECTION

The first step is to determine which forum technology a target website is running. This is a crucial step because it impacts how page navigation works (for example, phpBB vs. vBulletin have different URL schemes for threads and pages) [1]. The input to the LLM can be a snippet of HTML (such as the forum's front page or a thread page) or extracted textual parts from the page. In many cases, forums explicitly state their platform in the footer – e.g., "Powered by phpBB" or "Powered by SMF" as shown in Figure 1.

If such a signature exists, a simple automatic keyword check is sufficient. In the absence of explicit text, there are other indicative signs that can be used:

- **URL patterns**: phpBB URLs often include *viewtopic.php* or *viewforum.php* with parameters *f* (which stands for forum id) and *t* (for topic id). vBulletin 3/4 uses *showthread.php* and *forumdisplay.php*, whereas vBulletin 5 and some others use SEO-friendly paths but still might contain vbulletin in HTML comments or JavaScript code. SMF (Simple Machines Forum) typically uses *index.php?topic=* for threads and *board=* for sections, and Discuz! (a popular and widespread Chinese forum technology) uses URLs like *forum-<id>-1.html* and *thread-<tid>-<page>-1.html*. An LLM can be provided with a URL or HTML code snippet and prompted with the question: "Identify which forum software this site likely uses." Thanks to patterns seen during training, models like GPT-4 or Claude 2 can accurately classify the technology analysing forum software specific features. For example, experiments that were done in this work were in the form of feeding GPT-4 with the HTML head and part of the body of a phpBB forum page; the model correctly responded that the site was phpBB, noting the presence of phpBB specific features and typical structure.
- **HTML structure and keywords**: Each forum software has a default layout and often unique element IDs or class names. phpBB's HTML might contain references to classes like *postbody* or form fields with names like *sid* (session id) specific to phpBB. SMF pages show a distinctive table-like structure layout with user info sidebar and usually use labels like "Logged" under posts. Discuz pages might contain Chinese locale strings or certain script names. For the experiments in this work, a small prompt for the LLM known features of common forum technologies is compiled (e.g., "If you recognize 'post.php?action=post' and 'SMF' in the HTML,

Figure 1. An example of the explicitly stated platform of forum technology

it's SMF; if you recognize 'phpbb/templates', it's phpBB"). The idea of these types of prompts is to act as a few-shot guide. Upon providing the page HTML, the LLM outputs the guessed forum type with high confidence.

In cases where the forum is highly customized or built from scratch, the LLM might output "unknown/custom platform." This itself is useful feedback and it might also indicate that automated generation could be less reliable due to unrecognized structure. For such cases, the proposed approach defaults to a generic strategy (like SInFo [1] approach) or require human confirmation. In shown experiments, detection was straightforward on known forums – all three LLMs correctly identified the technology when clear signatures and software indicators were present. GPT-4 and Claude 2 even correctly identified the version of the forum technology analysing HTML structure indicators and noting "this is a phpBB 3.x forum", whereas the smaller Mistral model sometimes needed the explicit "Powered by" text to be sure.

3.2. SCRAPER CODE GENERATION

Once the forum type is identified, the prompt for the LLM to generate a scraper code is adapted to that specific forum technology. For each target technology, a distinct prompt template is designed, embedding general instructions on how to scrape forums. An example prompt (simplified) for phpBB is shown on Listing 1.

GPT-4 does not even requires a detailed template per platform – it often knows from its trainings the default behaviours. For instance, GPT-4 generated code that searched for a "Next" button or a *&start=* parameter for phpBB after seeing the page HTML code. However, to ensure reliability, the hints are provided in the prompt. For phpBB: "Note: phpBB thread pages use a *start=<n>* parameter for pagination. The first page might have *start=0* (implicit) and subsequent pages start=15, 30, ... etc., typically 15 or 20 posts per page. Use this knowledge to iterate through pages until no more posts can be found." For SMF: "SMF thread URLs contain *topic=<id>.<offset>*. E.g., *topic=123.0* for the first page, then *topic=123.15* for the next if 15 posts per page. Use the presence of a "next" link or increment the offset accordingly." By giving such technology-specific guidance, even a smaller model with fewer parameters like Mistral can follow the correct approach.

The LLM outputs code in Python, but any language could be requested, where Python is chosen for ease of readability and execution. The generated code typically includes: (1) sending an HTTP GET request to the initial URL, (2) parsing the HTML to extract the posts (using BeautifulSoup [13] or similar), (3) finding the URL or parameter for the next page, and repeating this process until no next page is found, and (4) storing or printing the extracted data (e.g., as JSON or CSV). The model is also instructed to include basic error handling (e.g., check response status, limited number of retries and break if a page request fails) to improve robustness.

When prompted for phpBB, GPT-4 correctly used the phpBB-specific classes (*postbody, author, content*) which it likely picked up from context or training knowledge. It also identified that the "Next" page link is a literal "Next" text anchor in phpBB default template. It is worth noting that GPT-4 was not prompted with specific class names – it reasoned them by itself, demonstrating the model's internal knowledge of phpBB HTML structure.

For models like Claude 2, the approach was similar. Claude tended to be very redundant with comments and sometimes over-engineered the solution (e.g., writing separate functions to parse a page). Compared to GPT-4, Claude also produced correct logic and even handled edge cases correctly like when a page has no "Next" link. On the other hand, Mistral 7B model, being a much smaller model, had experienced greater difficulty. Its first attempt at generating code often missed details (for example, it might not find the correct post container div, or it may stop after one page due to not recognizing the pagination element). By refining the prompt or providing an example HTML snippet to Mistral, it could correct itself to succeed in simple cases, but it was less reliable as out-of-the-box scraper code generation tool. This highlights a trade-off between large proprietary models and smaller open-source ones, which will be evaluated in the results.

```
"You are an expert web scraper. Write a Python script using requests and BeautifulSoup to
scrape all posts from a given phpBB forum thread URL. The script should handle pagination
by finding the 'next page' link or appropriate page parameters, and collect the author,
timestamp, and text of each post. Assume the forum is public (no login needed)."
```

Listing 1. An example of an LLM code generation prompt

3.3. EXECUTION AND DATA COLLECTION

After obtaining the code from the LLM, the next step is to run the scraper and collect the data. For this work and experiments, the generated script was manually executed to verify the correctness and to collect basic metrics. The focus of the evaluation was: Does the script successfully retrieve all posts from the forum (or thread)? And if so, how efficient is it (in terms of requests made or duplicates avoided)?

The advantage of having the forum exact type technology is the potential to optimize the crawling since the traversing structure can be known in advance. For instance, knowing the URL pattern, the LLM could decide to construct the URL for the last page of a thread (using information about total posts). SInFo did this by calculating the page index for the latest content [1]. Experiment with prompting GPT-4 to incorporate such logic in one case, was in the form of instruction "First fetch the thread's last page to get the latest posts, then retrieve earlier pages if needed and if they exist." GPT-4 responded with a complex approach which complicated the code. For simplicity and reliability, the main implementations in experiments remain regular forward pagination (page 1 → page 2 → ...). This returns all posts without needing to guess positions and is easier to validate for completeness.

It is important to note that while the LLM could in theory parse the HTML itself (without generating code) – as done in some end-to-end LLM scraping demos [12], having a concrete script offers persistence. The script can be reused to crawl the forum regularly, or shared, without requiring an LLM each time. Here, the LLM is observed as a "crawler generator". The main workload of the data extraction after that can be done by the code, which is efficient in execution, not the LLM. This addresses a common concern that LLM as a scraper might be slow or expensive if used for every page or frequently on a very large forum. Here, the LLM's cost is one time (per site) to produce a scraper code, comparable to a human developer writing it, but much faster.

3.4. LLMS COMPARED

Three different LLMs to the above tasks were observed on their differences: GPT-4, Claude 2, and Mistral 7B. Table 1 provides a summary comparison. GPT-4 (through OpenAI API, 8k context version) is the largest and generally most capable model that was tested in this work. Claude 2 (Anthropic model, ~100k context) is also a top-tier model with the notable advantage of a much larger context window. This in principle allows feeding it with a whole forum HTML page (tens of thousands of tokens) in one request. Mistral 7B is an open-source model released in late 2023. While not specifically a code model, in this work, the instruct variant was used with some success.

The prompting methodology was kept as consistent as possible. The same high-level prompts were used and only the details were adjusted necessary for each model (e.g., simplifying instructions for Mistral due to its smaller capacity). For GPT-4 and Claude, the temperature was set to 0 to minimize randomness, ensuring the outputted code is deterministic and focused. Mistral, being less deterministic even at low temperatures, sometimes generated different codes. The Mistral model ran a few times to take the best outcome for fairness.

Table 1. Comparison of LLMs for forum scraper generation

LLM Model	Size/Type	Speed (gen. tokens)	Code Accuracy	Context Window	Notes
GPT-4 (OpenAI)	~180B, Proprietary	~2-3 tokens/sec (API)	Very High – correct on first try for all tests< robust logic	~8K tokens (std. 32K variant available)	Best reasoning abilities; knows common forum patterns internally; slower and costlier.
Cloude 2 (Anthropic)	~120B, Proprietary	~5-7 tokens/sec (API)	Very High – correct in first try (small fix needed for one case)	100K tokens	Very large input capacity (good for long HTML); slightly more natural code comments; fast generation.
Mistral 7B (Open)	7B, Open-source model	~20+ tokens/ sec (local GPU)	Moderate – needed iterative prompting; prone to minor mistakes	~4K tokens (instruct variant)	Runs locally (no API needs); much weaker out-of-the box knowledge; can succeed with guided prompts.

4. EXPERIMENTS

Experiments were conducted to evaluate the accuracy and reliability of the LLM-generated scrapers, and to compare LLM performance in terms of speed and quality for this task. Representative forums were selected for each of the four target technologies (phpBB, vBulletin, SMF, Discuz). The detailed report was only on phpBB and SMF cases; the other two technologies were done in a more limited fashion due to time and scope of this paper, but the results were similar to phpBB and vBulletin.

For testing phpBB, the "Everything Search Engine" forum (voidtools.com/forum) was used. This forum statistics (as of March 2025) indicate over 21,000 posts in the support section alone, making it a robust test for scraping. For SMF, the "Fractal Softworks" forum was used (fractalsoftworks.com/forum), which is an SMF forum with ~109k posts in its general discussion board. Both forums are open access, no login is required to read. The approach was also verified on a smaller vBulletin forum and a Discuz! forum board with an English interface.

Testing pipeline was: feed an HTML snippet to LLM for detection, then prompt for code generation, then execute the code. The following were measured: (a) Was the platform correctly identified? (b) Did the code run without errors? (c) Did it successfully retrieve the expected number of posts? (d) How many HTTP requests did it use, and did it avoid unnecessary pages? And (e) time taken for the LLM to generate the code and for the code to run.

5. RESULTS

LLM Detection and code generation: All tested LLMs correctly identified the forum software in all cases where a clear signature was present (phpBB and SMF tests). For vBulletin, the HTML snippet that was provided was less explicit (the forum had removed the "Powered by" footer), but GPT-4 still reasoned it was vBulletin (likely from a meta tag and form field names), whereas Mistral misclassified it as "maybe phpBB or custom." This indicates that larger models have an advantage in hard to detect recognition tasks. Once the forum technology was known, GPT-4 and Claude 2 generated working code on the first try for both phpBB and SMF forums. Table 2 shows the scraping results. For the phpBB forum, the GPT-4 generated script successfully scraped all 4,008 threads and 21,239 posts from the "Support" board, by iterating through 40 pages of thread listings and then crawling each thread's pages. The entire crawl was completed in about 15 minutes. Claude 2 script was equally successful on phpBB. On the SMF forum, both models also managed to scrape the target board (General Discussion with ~7,906 topics and 109,942 posts). One minor issue arose on Claude for the SMF where initially generated code didn't construct the next page URL correctly (SMF requires adding an offset like .15 to the topic parameter). In both forums, Mistral 7B eventually produced a functional scraper but only after iterative prompting and guidance.

Speed and Efficiency: In terms of generation speed, GPT-4 was slower (it took ~30 seconds to output ~60 lines of code). Claude 2 was faster, producing similar length code in ~10 seconds. Mistral (running on a local machine with 1×A100 GPU in this test) was extremely fast in generating code – on the order of 20 tokens per second – but since it needed multiple tries, the total time to get the correct code was a couple of minutes. The scraping runtime for each script was comparable since they all used Python requests: the differences came down to how many requests were made. All scrapers successfully followed pagination and did not get dead locked in loops or miss pages. The number of page fetches roughly matched the number of pages in each forum section plus each thread. This is quite efficient given the scale, and importantly, it is the minimal required to get all posts.

Accuracy of Data: The scraped data were checked for errors. For instance, the first and last posts of certain threads were compared to the live website. In both cases, they matched exactly, including formatting. Both GPT-4 and Claude scrapers achieved 100% post recall on the tested forums. When finally worked Mistral code also got all posts from a couple of test threads, but it wasn't run on the full forum due to lower confidence in its generalization.

LLM Comparison: Table 1 summarizes the comparison among GPT-4, Claude 2, and Mistral 7B. GPT-4 demonstrated the highest reliability, consistently producing correct and well-structured code. It also tended to handle unexpected site anomalies better. Claude 2 advantages were its speed and extremely large context window. A big quality gap was not noticed between the models, both were excellent, with Claude making some minor mistakes. Mistral 7B clearly fell behind in understanding and needed more explicit instructions. Its main benefit is being open-source, so it can be self-hosted and fast.

Table 2. Results of LLM-generated scrapers on example forums

Forum (Sowtware)	LLM Usedfor Code	Post/Threads Retrieved	Page Fetched	Success?	Comments
Voidtools "Everything" forum (phpBB)	GPT-4 (Cloude 2)	21,239 posts in 4,008 threads (100% of forum section)	~8,000 HTTP requests (40 thread-list pages + thread pages)	Yes	Completed in ~15 min. Cloude 2 got same results. No duplicates: matched forum statistics.
Fractal Softworks forum (SMF)	GPT-4 (Cloude 2)	109,942 posts in 7,906 topics (full board)	~7,906 thread pages + navigation page (~8,200 total)	Yes	GPT-4 code handles SMF pagination correctly. Cloude needed one minor prompt adjustment then succeeded.
Tech Discussion forum (vBulletin 4)	GPT-4	~5,000 posts in 200 threads (sampled)	200 thread pages + index pages	Yes	Partial crawl (forum is large). LLM detection identified vBulletin. Scraper worked on first attempt.
Sample Discuzi forum (Discuz)	GPT-4	~500 posts in 50 threads (test sample)	50 thread pages + index pages	Yes	Discuz specific page pattern followed (thread *.html). Confirmed post count manually.
Overall		All public posts scraped with high accuracy	Efficiency near optimal		LLM-generated scrapers achieved complete data retrieval for each tested forum.

Table 2 highlights the end-to-end effectiveness on test forums. Overall, GPT-4 and Claude 2 are both excellent choices for implementing this type of scraping. Mistral 7B, while not as out of the box solution, is a promising sign that even lightweight models, which will only improve in coming years, could handle such tasks, especially if specialized code focused versions are used.

6. CONCLUSION

In this work, an LLM-based crawling approach is introduced that automates the detection and scraping of forums across technologies like phpBB, SMF, vBulletin, and Discuz, reducing manual effort. The results indicate that an LLM-driven approach to forum crawling is not only feasible, but also significantly influential. Unlike traditional crawlers, this method uses LLMs to generate customized scraping code based on forum structure. Experiments show that GPT-4 and Claude 2 produce accurate, efficient scrapers, while even smaller models like Mistral 7B can succeed with guidance. This lowers the entry barrier for web data collection and demonstrates the real potential of AI-assisted code generation. The development time for a new scraper is reduced from potentially days, if done manually, to minutes with proper LLM prompts. Moreover, the ability of LLMs to generalize well means that even if a forum theme or layout changes, the scraper code will still be correctly generated. The approach is extensible to other semi-structured web domains and opens paths for self-repairing, scalable scraping systems.

In conclusion, this experiment demonstrates that LLMs can serve as powerful allies in web forum crawling, automating what used to be time consuming engineering work.

REFERENCES

[1] M. Pavkovic and J. Protic, "SInFo – Structure-Driven Incremental Forum Crawler That Optimizes User-Generated Content Retrieval," *IEEE Access*, vol. 7, p. 126941–126961, 2019, doi:10.1109/ACCESS.2019.2939872.

[2] OpenAI, "GPT-4o," [Online]. Available: https://openai.com/index/gpt-4-research/. [Accessed 5 March 2025].

[3] Anthropic, "Claude 2," [Online]. Available: https://www.anthropic.com/news/claude-2. [Accessed 5 March 2025].

[4] M. AI, " Mistral 7B," [Online]. Available: https://mistral.ai/news/announcing-mistral-7b. [Accessed March 2025].

[5] W. Liu, Y. He, J. Wu, Y. Du, X. Liu, T. Xi, Z. Gan, P. Jiang and X. Huang, "A focused crawler based on semantic disambiguation vector space model," *Complex & Intelligent Systems*, vol. 9, no. 1, p. 345–366, 2023, doi:10.1007/s40747-022-00707-8.

[6] W. Huang, X. Li, X. Zhou, D. Qi, J. Xi, W. Liu and F. Zhao, "A Semantic and Optimized Focused Crawler Based on Semantic Graph and Genetic Algorithm," *Symmetry*, vol. 16, no. 11, p. 1439, 2024, doi:10.3390/sym16111439.

[7] J. Jiang, X. Song, N. Yu and C.-Y. Lin, "FoCUS: Learning to Crawl Web Forums," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, p. 1293–1306, 2013, doi:10.1109/TKDE.2012.56.

[8] reddit, "LLM Powered Web Scrapers Experience," [Online]. Available: https://www.reddit.com/r/LocalLLaMA/comments/1d5q6o7/llm_powered_web_scrapers_experience/. [Accessed 10 March 2025].

[9] "Scrapeghost," [Online]. Available: https://simonwillison.net/2023/Mar/26/scrapeghost/. [Accessed 25 March 2025].

[10] Firecrawl, "Turn websites into LLM-ready data," [Online]. Available: https://www.firecrawl.dev/. [Accessed 25 March 2025].

[11] LangChain, [Online]. Available: https://www.langchain.com/. [Accessed 25 March 2025].

[12] N. Corcuera Platas, "Enhancing Web Scraping With Large Language Models: A Modern Approach," 26 April 2024. [Online]. Available: https://medium.com/@ignacio.cplatas/enhancing-web-scraping-with-large-language-models-a-modern-approach-6216d5bba8d5#. [Accessed 25 March 2025].

[13] "beautifulsoup4," [Online]. Available: https://pypi.org/project/beautifulsoup4/. [Accessed 25 March 2025].