ChatGPT

Optimizing Small LLMs for HTML and Web Tasks: Literature Review and Thesis Positioning

Introduction

The rapid advancement of **large language models (LLMs)** has opened new possibilities for understanding and interacting with web content. The user's thesis goal is to **optimize a small LLM (≈7–8B parameters)** for tasks like **HTML understanding**, **data extraction from HTML**, and **intelligent web crawling**, with an emphasis on **local deployment** (running without cloud services). This report surveys academic publications and university theses (2020–2025) related to these topics, compares the user's goal with existing work, and analyzes the novelty of the proposed direction. We focus especially on theses from the **Faculty of Informatics and Information Technologies (FIIT), Slovak University of Technology** (FIIT STU) and similar works from other institutions.

We first review **prior research on LLMs for HTML understanding and web automation**, then summarize relevant **master's theses**. We compare these works to the user's thesis idea to identify overlaps and gaps. Finally, we suggest **refinements** to ensure the thesis offers a novel contribution, and we highlight any aspects the current goal may be missing relative to the literature.

LLMs for HTML Understanding and Data Extraction

Early approaches to web page understanding often relied on specialized architectures or heuristics to leverage HTML structure. For example, researchers developed bespoke models for form understanding or web navigation that required hand-crafted features or large annotated datasets [1]. However, recent work demonstrates that general-purpose LLMs can be **fine-tuned or prompted** to achieve strong results on HTML-based tasks without custom architectures [2].

Gur *et al.* (2023) showed that off-the-shelf LLMs, when fine-tuned, attain remarkable performance on HTML understanding tasks [3]. In their study, LLMs were evaluated on: (i) **Semantic Classification** of HTML elements (identifying the role of a given element, e.g. email input vs. password field), (ii) **Description Generation** (producing natural-language labels or instructions for an HTML form element), and (iii) **Autonomous Web Navigation** in a simulated browser environment [4] [5]. They found that a fine-tuned LLM could outperform prior specialized models – for instance, achieving a 50% higher task completion rate on a web navigation benchmark (MiniWoB) while using dramatically less training data (192× reduction) [3]. This indicates that even **moderate-sized LLMs** can learn HTML structures and user interface semantics efficiently through transfer learning. Notably, their experiments included models ranging from ~24M up to 62B parameters [6], suggesting that **scaling down** to smaller LLMs is possible, though the paper implies some performance trade-off with very small sizes (exact 7B–8B results are not explicitly stated in the excerpt).

Another line of work integrates **HTML's structural information directly into model training**. For example, **StructuralLM** and **MarkupLM** (2021) introduced pre-training techniques to imbue transformers with awareness of markup structure for documents and web data [7] [8]. These models (based on the Transformer architecture) incorporate HTML tags and DOM tree positioning into the input, enabling better performance on tasks like form understanding and web information extraction.

Such approaches can be seen as optimizing smaller language models by **specialized pre-training** on HTML data, rather than relying purely on generic LLM pre-training. They demonstrate that models far smaller than GPT-3 can still be effective in web data extraction if trained appropriately with structural cues.

In 2023, **Li *et al.*** proposed **WIERT (Web Information Extraction via Render Tree)**, a novel transformer-based model for web page data extraction [9]. WIERT leverages the **render tree** (a combination of the DOM tree and CSS styling) to encode both semantic text content and visual layout information of a page [10]. By using a pre-trained language model as a base and injecting render-tree features, WIERT effectively captures context that a plain-text representation would miss (such as which text is visually grouped under which heading or in which table). This approach does not explicitly focus on model size, but it shows another **optimization strategy for web data extraction**: enhancing input representations (which could benefit a smaller LLM by making relevant features more accessible). The success of WIERT indicates that a thesis could explore using **HTML rendering or structural parsing to aid a 7B LLM**, improving its understanding of web pages without needing billions of parameters.

Beyond static understanding, recent research tackles **LLM-driven web interaction (crawling and form-filling)**. An arXiv study by **Huang *et al. (2024)*** introduced **AutoCrawler**, a two-stage LLM-based framework for intelligent web crawling [11]. The idea is to have the LLM **generate a crawler program** (a sequence of DOM actions or extraction rules) for a given vertical domain, rather than manually writing scraping rules. AutoCrawler uses the hierarchical HTML structure in a "top-down and step-back" process, where the LLM progressively explores the DOM, learns from mistakes, and refines its actions [12] [13]. This approach addresses two challenges observed with naive LLM web agents: low success rates on complex sites, and poor reusability (an LLM starting from scratch on each page) [14] [15]. By generating intermediate structured rules, AutoCrawler's agents became more efficient and could handle changing websites better than pure prompt-based agents. While the paper primarily tested larger LLMs and multiple models in an ensemble, the paradigm is relevant – a smaller LLM could be fine-tuned or instructed to produce scraping **actions** (like XPath queries or form submissions) by understanding the HTML, then those actions can be executed by a lightweight script. This combination of **LLM reasoning with traditional crawler execution** could be a fruitful direction for the user's project, as it introduces structure and memory that purely generative agents lack. It's worth noting that AutoCrawler's motivations align with the user's goal: improving adaptability and efficiency of web crawlers via LLM understanding of HTML.

In the realm of **web form interaction**, **Chen *et al. (2025)*** explored using LLMs to automate form filling for web application testing [16]. They augmented a prior reinforcement learning–based tester with an LLM to generate realistic input data and to help determine if a form submission was successful. Specifically, they fine-tuned Google's T5 model (a relatively small transformer model) with *prompt tuning* to classify the type of each form field (e.g., name, email, date) and then used a data faker to generate appropriate inputs [16] [17]. They also partially relied on an LLM (referred to as GPT-4^o) to decide if the result page indicated a successful submission [16]. By integrating these LLM components, the system achieved better code coverage in testing (2.3% higher than the RL-only model, and up to ~12% better than another baseline) [18]. This work highlights how **prompt optimization and fine-tuning on specific sub-tasks** (field classification in this case) can improve a small model's effectiveness in an HTML context. It also underscores an important point for the thesis: sometimes *partial use of an LLM* for the intelligent parts (like understanding context or validating outcomes) combined with traditional automation can yield the best results. A small model might handle classification or extraction, while the crawler does clicking and form submission, achieving a balance between local computation and task complexity.

In summary, the literature shows **multiple strategies to optimize smaller LMs for HTML tasks**:

- Fine-tuning LLMs on structured web tasks to leverage transfer learning [3].
- Pre-training or augmenting models with HTML/DOM structural signals [8] [9].
- Using LLMs to *guide* or *generate* extraction rules and actions for crawlers (combining learning with deterministic execution) [11].
- Applying prompt engineering or lightweight tuning (e.g. prefix prompts, LoRA, etc.) for domain-specific needs like form field understanding [16].

These works largely validate the feasibility of the user's direction, but also illustrate potential pitfalls (context length issues with raw HTML, the need for structural awareness, etc.). Notably, many high-profile studies (Google's, Fudan's) involved models larger than 7–8B or used proprietary LLMs; however, the techniques they propose could be **adapted to a 7B scale**. The user's thesis can carve a niche by explicitly focusing on **resource-constrained scenarios**—for instance, running a fine-tuned **7B model on local hardware** and measuring how well it matches the performance of larger models on these HTML tasks.

LLM-Driven Web Agents in Academic Theses

Academic **master's theses** in recent years have also started exploring small-scale LLM applications for web data, often motivated by practical deployment needs (offline use, specific industry applications). Below we discuss several relevant theses, highlighting their goals, methods, and how they intersect with the user's proposed work.

- **Matej Kollár (FIIT STU, 2025)** – *"Local LLM Chatbot for AiA Platform"*. This thesis dealt directly with deploying an LLM solution **without Internet or cloud services**, aligning with the user's local deployment focus. Kollár designed a chatbot for an enterprise platform (AiA by Itaps s.r.o.) that lets employees query internal documents in natural language [19]. The solution combined a local LLM with a **Retrieval-Augmented Generation (RAG)** approach: documents were indexed in a vector database, and relevant passages were retrieved to ground the LLM's answers [20] [21]. By pairing a small transformer-based model with efficient semantic search (FAISS or similar), the system provided accurate answers while keeping data on-premises. This thesis underscores a key optimization strategy for small models: **compensating smaller size with external knowledge retrieval**. Rather than relying on a giant model's parametric knowledge, a 7B model can be boosted by retrieving exact information from HTML pages or databases. The user's goal of HTML data extraction could similarly use RAG – e.g., first scrape and chunk the HTML content, use an embedding model to find relevant pieces, then have the LLM interpret or summarize those pieces. Kollár's successful defense of this project [22] [19] suggests that a well-engineered local LLM pipeline can meet real-world requirements, an encouraging sign for the user's thesis viability.

- **Simon Kokavec (FIIT STU, 2025)** – *"Impact of LLM and AI on the Real Estate Market: Replacing Human Agents in Searching for Properties"*. Kokavec built an **autonomous agent** that helps users search for real estate listings using natural Slovak language [23]. The system used **modern LLMs combined with vector-based semantic search**, allowing users to input detailed queries (e.g. "quiet 2-bedroom apartment near a park under €700/month") instead of clicking through rigid filters [24] [25]. The thesis details an end-to-end development: web scraping real estate listings, indexing them in a vector database, and constructing a chat-based interface where the LLM interprets the query and retrieves relevant results [26]. Through experiments, Kokavec showed that this natural-language approach improved the user experience and search accuracy compared to traditional methods [27] [28]. This thesis is very relevant: it blends **web data**

extraction (scraping listings HTML), **small LLM usage**, and **local deployment** (likely using an open-source Slovak or multilingual model for on-premise use). It demonstrates that even without a massive model, one can leverage embeddings and LLM reasoning to build an "intelligent crawler" that understands user intent. The user's thesis could take inspiration from this by, say, focusing on a different domain (not real estate but perhaps e-commerce or academic content) and ensuring the small LLM is optimized (via fine-tuning or prompt engineering) for that domain's HTML layout peculiarities.

- **Marek Chrappa (FIIT STU, 2025)** – *"AI in Customer Support: Extending and Replacing the Human Role"*. This thesis provides a broad overview of using AI (especially NLP and LLMs) in customer service contexts [29]. It doesn't focus solely on HTML or crawling, but it does discuss real-time interactions and the integration of voice/text systems with LLMs [30]. Chrappa presents case studies like chatbots and multimodal assistants, evaluating factors such as response quality, cost, and user experience [31]. One point highlighted is the **challenge of integrating LLMs in production**, including concerns of **latency and trust** [32] [33]. This is tangentially relevant: if the user's small LLM is to be used in an intelligent web crawler, considerations like speed (small models are generally faster, but if not optimized could still be a bottleneck when parsing thousands of pages) and reliability (users not fully trusting AI outputs without verification [33]) will arise. Chrappa's work suggests emphasizing the **human-in-the-loop or validation mechanisms** in the thesis, to address the skepticism around LLM decisions. For instance, an intelligent crawler might use the LLM to suggest which pages to crawl or which data to extract, but also log reasoning or confidence so a developer can review its decisions – thus improving trust.

- **Tomáš Ondruš (FIIT STU, 2025)** – *"Detection of Cybersecurity Incidents Using LLMs"*. Ondruš's thesis (approved June 2025) applied LLMs to analyze cybersecurity events and incidents [34]. While details are sparse in the summary, it involved training and using an LLM (specifically mentions "LLaMA 3.2" in the FIIT library record) to detect security issues, and deploying the model in a Docker container for local use [35]. The key takeaway for our context is that this is another example of **fine-tuning a smaller open-source LLM** for a niche task (security logs or incidents) and packaging it for local inference. It reinforces the trend that **7B–13B parameter models** (e.g., LLaMA 2, etc.) are becoming common in theses for domain-specific applications. The optimization often involves curating fine-tuning data and possibly applying efficient finetuning methods (LoRA, QLoRA) to adapt the model on limited hardware. Ondruš's project likely had to address **data preparation (converting logs or incidents to a format the LLM can learn from)** and **evaluation of accuracy** – similarly, the user's thesis will need to gather training data for HTML tasks (perhaps using public datasets like MiniWoB [36] [37], or auto-generated data from Common Crawl as Gur *et al.* did [38] [39]) and decide on suitable metrics (accuracy of extraction, success rate of navigation, etc.).

- **Tereza Tódová (Masaryk University, 2025)** – *"A Quest for Information: Enhancing Game-Based Learning with LLM-Driven NPCs"*. This award-winning thesis explored using LLMs (notably a local **LLaMA-based model**) to power interactive non-player characters in an educational VR game [40] [41]. The NPCs could engage in open-ended dialogue with players, providing hints and information to help with quests. Tódová's focus was on making the solution **"low-cost and easily extendable"** [42] – meaning the LLM had to run efficiently on available hardware and be adaptable to new dialogue scenarios without retraining from scratch. She compared various approaches to **customize LLM behavior** for the game world, which likely included prompt engineering vs. fine-tuning experiments [41] [43]. For example, one could imagine she tried methods like few-shot prompting the model with game context, or fine-tuning on Q&A pairs relevant to the game lore, or using **embedding-based retrieval** of game facts (similar to RAG) so

that a small model can output accurate story-related info. This thesis is relevant in that it explicitly addresses **optimizing a small LLM for a specialized task with local deployment**. It shows the importance of *prompt design* – the thesis likely had to craft prompts that keep the NPCs "in character" and on topic, which parallels how an HTML-understanding LLM might need careful prompt formatting to steer it (e.g., telling the model to output JSON extracted from an HTML, or to think step-by-step when navigating). The **findings that came out of comparing customization approaches** (e.g., prompt templates, system messages, fine-tuning on dialogue data, etc.) can inform the user's project. They might discover, as Tódová did, that a bit of fine-tuning drastically improves consistency of a small model in a niche domain, or conversely that clever prompting is enough in some cases. Incorporating such lessons could strengthen the thesis by not just doing *something* with a 7B model, but doing it in a principled way that builds on known best practices.

- **Qiaoqiao Zou (Häme University of Applied Sciences, 2025)** – *"Designing a Smart Search Pipeline with Web Scraping and LLM-based Retrieval on the HAMK Website"*. This thesis from Finland tackled a problem very close to "HTML understanding + local LLM." Zou built a system to improve information access on a university website by combining **web scraping, semantic embeddings, and LLMs** [44] [45]. The HAMK site was static and fairly structured, yet users had difficulty finding specific content via manual browsing [46]. The implemented pipeline automatically crawled the site (using Scrapy) to **extract pages and content**, then used **SentenceTransformer embeddings** to index the content, and finally employed an LLM to answer user queries by retrieving and summarizing relevant information [45] [47]. The interface was built with Gradio for user interaction [48]. Essentially, this is a **local Q&A chatbot over a scraped HTML corpus**, using a small LLM and vector database – almost exactly the kind of solution one might propose for an intelligent documentation search or a domain-specific web assistant. The thesis is highly relevant because it demonstrates that **even without giant models, effective web data extraction and QA is achievable by pipelining technologies**: a focused crawler, a good embedding model for semantic search, and an LLM to generate answers [47]. For the user's goal, this suggests a possible architecture and underscores the value of **modular optimization** (each part of the pipeline can be optimized independently – e.g., tuning the crawler to get clean HTML, tuning the embedding model to handle HTML text better by perhaps including HTML tags in the text it embeds, and prompt-tuning the LLM to produce concise answers). Zou's work likely used a reasonably small LLM (the thesis doesn't specify, but given the context it might have been GPT-J or an instruction-tuned 7B model, or even an API like OpenAI if allowed – though the emphasis on *local* tools like FAISS and Gradio suggests open-source). The user's thesis could take a similar approach but perhaps extend it with **more "intelligent" crawling** – e.g., deciding which pages to prioritize by analyzing the HTML with the LLM (this goes a step beyond Zou's pipeline, which treated all pages fairly). That would introduce novelty by adding an LLM-based **crawler decision-making** component to an otherwise similar setup.

- **Dylan Elens (Radboud University, 2025)** – *"An LLM-Based End-to-End Testing Agent for Web Applications"*. In this thesis, Elens implemented and evaluated an LLM-driven agent that performs end-to-end UI testing on web apps (essentially using an LLM to simulate a human tester in a browser) [49] [50]. The findings were mixed: the LLM agent could handle basic web elements and follow natural-language test plans, but struggled with highly interactive components (e.g. complex JavaScript widgets) [32]. Developers liked the concept of autonomous testing but were concerned about the **slow speed and reliability** of the LLM-based approach [33]. This highlights two important considerations for any intelligent web crawler using LLMs: **performance optimization and error handling**. Small LLMs are faster than large ones, but running an LLM in a loop for many pages could still be slow. Elens' work suggests perhaps using the LLM only for the hard parts of the task and using simpler scripted logic for the rest. It also underlines the

need for a crawler to handle cases the LLM fails on (time-outs, unexpected formats). For the user's thesis, learning from these limitations will be crucial – e.g., incorporating a fallback strategy when the LLM is not confident, or limiting the scope of tasks given to the LLM to those it can do reliably (maybe parsing static content is fine, but a small LLM might not reliably execute complex form sequences without fine-tuning on those exact sequences).

Comparison and Novelty Analysis

Overlap with Existing Work: The user's thesis goal – small LLM optimized for HTML understanding, extraction, and crawling – clearly intersects with multiple lines of current research. As seen above, there are **published methods and thesis projects that use LLMs for:**

- Parsing HTML to identify elements or extract data (Gur et al. with fine-tuned LLMs [4] [3], WIERT with a transformer for web IE [9]).
- Navigating web pages or filling forms autonomously (AutoCrawler's LLM-guided actions [11], RL+LLM for form filling [16], WebGPT and related agents as referenced in literature [51] [52]).
- Domain-specific information retrieval from web content using local LMs (Kollár's and Zou's RAG pipelines [20] [47]).

This means the thesis is **not entirely novel in objective** – many researchers recognize the opportunity to apply LLMs to web data problems. If the user proceeds without adjusting scope, there's a risk of overlapping significantly with prior work, especially the **Google 2023 HTML understanding** study and the **AutoCrawler 2024** framework. For example, if the thesis was simply "fine-tune LLaMA-7B on MiniWoB to navigate websites," that would be a smaller-scale replication of Gur et al.'s work [5]. Likewise, a project to "use GPT-4 via API to build a web scraping agent" would overlap with known systems like WebGPT (Nakano et al. 2021) and WebAgent (Deng et al. 2023) [14].

However, **the novelty can be established** with careful focus on what others *haven't* done or haven't solved well:

1. **Emphasis on Small Models & Local Deployment:** Many academic papers used either very large models or did not constrain themselves to running locally. The thesis can specifically contribute knowledge on how to make a **7B model as effective as possible** for these tasks, detailing the trade-offs and techniques needed. For instance, the thesis could experiment with **compression or quantization** (like 4-bit quantization of the model weights) to allow the model to run on edge devices, and measure the impact on accuracy. This would be a practical addition to literature that often assumes access to vast compute. Collating best practices for fine-tuning a small LLM on an average GPU, or using **LoRA adapters** to cheaply train it on HTML data, would be very useful to the community of developers who cannot run 70B models. The FIIT STU context, where the student actually implements and deploys locally (as seen in Kollár's and Ondruš's theses), ensures real-world validation of this approach.

2. **Integrating Structured Knowledge or Tools:** The thesis could stand out by how it enhances the small LLM's abilities. For example, implementing an **HTML pre-processing step** where the HTML is cleaned, pruned, or structured before feeding to the model could be novel. The recent HtmlRAG paper (Wang et al. 2024) pointed out that feeding raw HTML (with all the CSS/JS) into an LLM wastes context on irrelevant tokens and proposed cleaning and chunking strategies [53] [54]. A 7B model with only a 4K or 8K token context would greatly benefit from such HTML cleaning [55] [54]. If the user experimentally develops a **pipeline that strips boilerplate, retains meaningful tags (links, tables, form fields)**, and perhaps even converts some of the DOM hierarchy into hints (like prepending "Section: XYZ" headings in the text), that would be a useful

contribution not deeply covered in prior theses. Similarly, using **external tools in the loop** – e.g., a JavaScript renderer for dynamic pages, or a simple rule-based parser for tables that works with the LLM – could improve performance and show a novel hybrid approach (LLM + traditional methods) optimized for local use.

3. **New Application Scenario:** The thesis could introduce a scenario or dataset that hasn't been extensively studied with small LLMs. For example, an **"intelligent crawler" for a specific domain** (say, academic papers on the web, or government open data sites) where the crawler must understand semi-structured HTML pages and extract key facts. By focusing on a niche, the thesis ensures it's not just rehashing generic benchmarks. Any unique findings (e.g., "Model X struggled with tables until we fine-tuned on only table-tag data") would add to the body of knowledge. It would also let the student generate their own evaluation metrics tailored to the domain (perhaps combining precision/recall of data extraction with crawl efficiency metrics).

4. **Evaluation of Optimization Strategies:** The thesis can compare different ways to optimize a small LLM for HTML tasks – something that is of interest to many but not systematically analyzed in literature. For instance, **prompt design vs. fine-tuning**: Tódová's game NPC work suggests exploring both, and a careful evaluation could reveal when a cleverly crafted prompt suffices and when a fine-tuned model is necessary for HTML understanding. Or, comparing a model pre-trained on code (which might handle HTML syntax better) versus a model pre-trained on pure natural language for this task. If the user finds, say, that a 7B **CodeLLM (like CodeGen or StarCoder)** outperforms a 7B general LLM on parsing HTML DOM trees, that's a valuable insight to report. Likewise, analyzing the effect of adding HTML tag tokens to the model's vocabulary or input could be new.

5. **Addressing Weak/Missing Aspects:** Based on our literature survey, a few aspects seem underrepresented and could be opportunities for the thesis:

6. **Robustness and Errors:** How to ensure the small LLM doesn't hallucinate or misinterpret HTML content. The thesis can incorporate validation steps (like schema checks for extracted data) or confidence estimation (e.g., have the LLM explain its extraction and verify key points). This directly addresses the trust concerns noted by developers in Elens (2025) [33] and Chrappa (2025).
7. **Speed Improvements:** Using a small model is one way to improve speed; additionally the thesis could implement batching of multiple extraction queries, or using the LLM only on sampled pages rather than all, to show a **resource-conscious crawling strategy**. AutoCrawler's idea of reusability is relevant here – if the LLM "invents" an XPath rule to get product prices from an e-shop, that rule can be reused on many pages quickly without querying the LLM each time. Demonstrating this kind of caching or rule learning with a small model would be quite novel.
8. **Multimodal Integration:** Most current works either treat it as text-only (HTML as text) or go fully visual (render tree). Perhaps a middle ground could be explored: e.g., use the LLM to decide if a screenshot is needed (for a particularly complex layout) and then use an OCR or vision model. Or feed the LLM pseudo-visual cues (like `[IMAGE]` tags where images are, or lengths of content). While a full exploration might be beyond scope, acknowledging HTML's visual aspect (as WIERT does) [10] could make the thesis more comprehensive.
9. **Security and Ethics:** If crawling the web intelligently, one should consider robots.txt, rate limiting, and data privacy. The thesis can distinguish itself by embedding ethical considerations – for instance, ensuring the intelligent crawler respects crawling policies or discussing how an LLM might inadvertently extract sensitive info if not constrained. This is somewhat beyond pure ML optimization, but it's a **missing discussion in many technical papers** that a thesis could

address, making it a more rounded contribution (especially important if publishing or applying the results in real settings).

In conclusion, the user's thesis idea is **timely and has a fertile ground of prior work to build upon**, but to be **novel** it should not just apply an existing large-model solution at smaller scale. It should instead highlight new insights or methods for making small LLMs successful in HTML and web tasks. The literature suggests focusing on *efficient fine-tuning, clever use of HTML structure, integration with retrieval and tools, and thorough evaluation*. By doing so, the thesis can confidently claim a **new approach or direction** – for example, *"demonstrating a locally fine-tuned 7B model that can navigate and scrape websites with X% of the accuracy of GPT-4 at a fraction of the computing cost, through method Y"*, or *"introducing a hybrid pipeline that outperforms end-to-end LLM prompting on web extraction by combining rule learning and LLM inference."* These would be worthwhile contributions that overlap minimally with existing publications.

The following table summarizes and compares key related works to the user's thesis concept, illustrating where the thesis can differentiate itself:

Publication / Thesis (Year)	**Focus & Task**	**Model & Optimization**	**Overlap with User's Goal**	**Novelty / Gaps**
Gur et al. (Google DeepMind) (2023) [4] [3]	HTML understanding (form fields, description, navigation)	Fine-tuned LLMs (24M–62B parameters) on HTML tasks	Shows LLMs can parse HTML and even navigate autonomously	Used larger models; user can try to replicate results with a **7B model**, focusing on local deployment and efficiency.
Huang et al. – AutoCrawler (2024) [11] [12]	Intelligent web crawler generation (vertical domains)	Prompted LLM agents + learning from mistakes	Combines LLM reasoning with rule generation for crawling	Mainly conceptual; user can implement a similar idea on a small model and report performance, and extend it with **reusable rules for speed**.
Chen et al. (MDPI) (2025) [16]	Web form filling for testing	Used T5 (prompt tuning) + GPT-4 (partial)	Uses LLM to understand fields and generate inputs	Narrow subtask overlap (form understanding); thesis can generalize this to **broader HTML element classification** with a small LLM, and handle form submission decisions locally.

Publication / Thesis (Year)	**Focus & Task**	**Model & Optimization**	**Overlap with User's Goal**	**Novelty / Gaps**
WIERT (Li et al.) (2023) [9] [10]	Web info extraction with visual context	Transformer with render-tree encoding (pretrained LM)	Both aim to extract structured data from pages	WIERT adds visual features; user's LLM could not handle images, but could mimic some layout awareness via HTML tags. A possible new angle is to incorporate **CSS-based cues or lightweight vision** into a small LLM pipeline.
Kollár (FIIT STU) (2025) [19] [56]	Local LLM Q&A on internal documents (RAG approach)	Open-source LLM + FAISS vector DB, all on-premise	Emphasizes offline LLM use and retrieval augmentation	Directly relevant architecture; thesis can adopt RAG for web data. Novelty comes by applying it to live web crawling and focusing on HTML-specific challenges (Kollár dealt with static docs, not crawling).
Kokavec (FIIT STU) (2025) [26] [28]	Natural-language real estate search agent	Embeddings for listings + LLM for dialogue	Shows LLM-guided search on crawled data	Similar in concept (semantic search over crawled HTML). The thesis can differentiate by targeting a different domain and by fine-tuning the LLM on that domain's terminology/HTML patterns for better accuracy.

Publication / Thesis (Year)	**Focus & Task**	**Model & Optimization**	**Overlap with User's Goal**	**Novelty / Gaps**
Tódová (Masaryk U.) (2025) [40] [41]	LLM-driven NPC dialogues in VR game	LLaMA-based model, low-cost, custom prompt tuning	Optimizing small LLM behavior in interactive setting	Overlap in methodology (prompt vs fine-tune comparisons). User can use her insights on **LLM customization** to inform HTML-task prompt design. A novel contribution would be a systematic evaluation of prompt engineering techniques for HTML parsing (something game NPC work touches tangentially).
Zou (HAMK) (2025) [45] [47]	QA chatbot over university website content (static crawl)	Scrapy + SentenceTransformer + local LLM (Gradio UI)	End-to-end pipeline for web content Q&A	Very similar pipeline. The thesis must go beyond QA on static content – e.g., introduce **active crawling (deciding which new pages to visit based on LLM analysis)**, or optimizing the pipeline for incremental updates, etc., to be novel.

Publication / Thesis (Year)	**Focus & Task**	**Model & Optimization**	**Overlap with User's Goal**	**Novelty / Gaps**
Elens (Radboud U.) (2025) [50] [32]	LLM-based web UI testing agent	GPT-3.5 or similar agent executing tests in browser	LLM interacting with live web interface	Highlights issues of speed and reliability with LLM agents. Thesis can address these by choosing a smaller model and adding fallbacks. A new angle: measuring how **quantization or smaller model size improves speed** for web crawling tasks.

Table: Comparison of key related works with the user's thesis idea. (**Bold** text indicates particularly relevant aspects). The user's thesis can draw on these works for inspiration but should contribute new findings on **small-model optimization, novel combinations of techniques, or a unique application domain** to ensure an original contribution. Citations indicate source of information for each work.

Conclusion and Recommendations

The review above demonstrates that while the core idea of using LLMs for HTML understanding, extraction, and web automation is an active area of research, there remains ample room for the user's **7B-scale LLM project to be novel**. To ensure the thesis introduces a new approach or insight, we recommend the following refinements:

- **Highlight the "small LLM" angle:** Make the constraints a centerpiece – e.g., *"How far can we push a 7B model in web understanding tasks, and what optimizations are necessary to reach acceptable performance?"* This quantitative angle (perhaps comparing a 7B to a 13B or showing how optimizations close the gap) will be interesting to document, since many publications don't explicitly focus on resource-limited scenarios.

- **Incorporate Structural Aids:** Plan to utilize HTML structure in clever ways (DOM parsing, cleaning, segmentation) to assist the LLM. This could be a rule-based preprocessor or a learned component (even a smaller model that labels parts of the page for the main LLM to consume). By doing this, the thesis stands out from those that treated the LLM as a black box and shoved raw HTML at it. It plays to the strength of having full control in a local setup.

- **Combine Methods for a Pipeline (Hybrid System):** Rather than only fine-tuning or only prompting, consider a pipeline where the crawler and LLM each do what they're best at. For example, use deterministic scripts to gather candidate links or form fields, use the LLM to choose among them or extract meaning, then script to output in a structured format. This hybrid approach would be novel compared to end-to-end LLM attempts, and it's practical for local deployment (since you minimize LLM calls). Documenting the design rationale of such a system would be a valuable guide for others.

- **New Evaluation Benchmarks:** If possible, create a small benchmark relevant to the thesis (or adapt an existing one) to evaluate the system. For instance, measure how accurately the model extracts a set of fields from a variety of websites (there are datasets like SWDE for structured web data extraction, which could be used [57]). Or measure how many pages the intelligent crawler can process per minute using a 7B vs a 70B model. Providing these empirical results will highlight the thesis's contribution – e.g., *"First evaluation of a LLM-based crawler on X dataset using a model under 10B parameters."* Even negative results (if the small model fails on some aspects) are informative and can be turned into discussion points on limitations and future work.

- **Address Weaknesses Proactively:** Plan for error cases – maybe implement a simple verifier that checks if extracted data makes sense (and if not, maybe fall back to a simpler regex extract, or mark it for review). This kind of robustness is often missing in research prototypes and can be a selling point of the thesis as a **complete, deployable solution**.

By following these recommendations, the user's thesis will position itself at the frontier of *applied* LLM research: pushing the envelope of what **small, locally-run language models** can achieve in understanding and interacting with the web. It will be novel in combining insights from multiple strands of research (NLP, web mining, information retrieval) and catering them to a constrained setting. Ultimately, the work can produce a blueprint for others looking to build **cost-effective, privacy-preserving intelligent web agents**, which is a genuine gap in the current literature where the trend often tilts towards ever-larger, cloud-based models.

---

[1] [2] [3] [4] [5] [6] [7] [8] [36] [37] [38] [39] [51] [52] aclanthology.org
https://aclanthology.org/2023.findings-emnlp.185.pdf

[9] [PDF] WIERT: Web Information Extraction via Render Tree
https://ojs.aaai.org/index.php/AAAI/article/view/26546/26318

[10] WIERT: web information extraction via render tree
https://dl.acm.org/doi/10.1609/aaai.v37i11.26546

[11] [12] [13] [14] [15] [57] AutoCrawler : A Progressive Understanding Web Agent for Web Crawler Generation
https://arxiv.org/html/2404.12753v1

[16] [17] [18] Using Large Language Model to Fill in Web Forms to Support Automated Web Application Testing | MDPI
https://www.mdpi.com/2078-2489/16/2/102

[19] [20] [21] [56] Final theses
http://is.stuba.sk/zp/portal_zp.pl?prehled=vyhledavani;podrobnosti=184400;zp=100930;dinfo_jazyk=3

[22] Final theses
http://is.stuba.sk/zp/portal_zp.pl?podrobnosti=184400

[23] [24] [25] [26] [27] [28] Final theses
http://is.stuba.sk/zp/portal_zp.pl?podrobnosti=184121

[29] [30] [31] Final theses
http://is.stuba.sk/zp/portal_zp.pl?podrobnosti=183863

[32] [33] [49] [50] cs.ru.nl
https://www.cs.ru.nl/bachelors-theses/2025/Dylan_Elens___1051438___An_LLM-Based_End-to-End_Testing_Agent_for_Web_Applications.pdf

34 Detection of cyber security incidents using Large Language Model
https://kis.stuba.sk/arl-stu/en/detail-stu_us_cat-stuzp105884-Detection-of-cyber-security-incidents-using-Large-Language-Model/?disprec=1&iset=1

35 Final theses
http://is.stuba.sk/zp/portal_zp.pl?podrobnosti=184059

40 41 42 43 Thesis/Dissertation: Bc. Tereza Tódová: A Quest for Information: Enhancing Game-Based Learning with LLM-Driven NPCs
https://is.muni.cz/th/wwa0n/?lang=en

44 45 46 47 48 theseus.fi
https://www.theseus.fi/bitstream/10024/894868/2/Zou_Qiaoqiao.pdf

53 54 55 HtmlRAG: HTML is Better Than Plain Text for Modeling Retrieved Knowledge in RAG Systems
https://arxiv.org/html/2411.02959v1