Knowledge Extraction on Semi-Structured Content: Does It Remain Relevant for Question Answering in the Era of LLMs?

Kai Sun[1], Yin Huang[1], Srishti Mehra[1], Mohammad Kachuee[1], Xilun Chen[2], Renjie Tao[1], Zhaojiang Lin[1], Andrea Jessee[1], Nirav Shah[1], Alex Betty[1], Yue Liu[1], Anuj Kumar[1], Wen-tau Yih[2], Xin Luna Dong[1]

[1]Meta Reality Labs, [2]FAIR, Meta

Correspondence: sunkaicn@meta.com

Abstract

The advent of Large Language Models (LLMs) has significantly advanced web-based Question Answering (QA) systems over semi-structured content, raising questions about the continued utility of knowledge extraction for question answering. This paper investigates the value of triple extraction in this new paradigm by extending an existing benchmark with knowledge extraction annotations and evaluating commercial and open-source LLMs of varying sizes. Our results show that web-scale knowledge extraction remains a challenging task for LLMs. Despite achieving high QA accuracy, LLMs can still benefit from knowledge extraction, through augmentation with extracted triples and multi-task learning. These findings provide insights into the evolving role of knowledge triple extraction in web-based QA and highlight strategies for maximizing LLM effectiveness across different model sizes and resource settings.

1 Introduction

Over the past decade, Knowledge Graphs (KGs) have played a pivotal role in advancing Question Answering (QA) systems, powering major industrial applications such as Google and Bing's search (Zou, 2020). Closely tied to this progress is the field of Information Extraction, which focuses on deriving structured knowledge—typically in the form of triples—from unstructured or semi-structured text (Srihari et al., 1999; Khot et al., 2017). Recently, Large Language Models (LLMs) have shown remarkable proficiency in interpreting unstructured content and generating accurate responses directly from it, a capability central to techniques like *Retrieval-Augmented Generation (RAG)* (Lewis et al., 2020). In light of these developments, an important question arises: *in an era dominated by powerful generative models, does the task of structured knowledge extraction still hold relevance for question answering?*

In this paper, we investigate a specific category of web content: *semi-structured* webpages, where information is presented with some underlying structure such as web tables or attribute–value pairs, rather than in natural language texts. Our focus on semi-structured webpages is motivated by two key factors. First, the web contains a wealth of semi-structured data, typically populated by data from large underlying databases, thus offering a valuable source of factual knowledge (Dong et al., 2014). Second, it remains uncertain whether LLMs are equally adept at handling semi-structured content, which is rich in format but often lacks the grammatical signals to guide language understanding.

Knowledge extraction from semi-structured webpages has been extensively studied in the literature (Lockard et al., 2020, 2019, 2018; Gulhane et al., 2011; Kushmerick et al., 1997). These studies highlight the unique challenges and opportunities posed by semi-structured content, differentiating it from extraction on unstructured texts (Lockard et al., 2018). Despite the demonstrated high quality for the ClosedIE setting, where attributes are pre-defined and training examples abound, OpenIE, where attributes and even domains are unknown, remains difficult, with performance capped at a 46% F-measure (Lockard et al., 2020). This paper investigates three key questions to assess the value of knowledge extraction for QA on semi-structured data.

RQ1: How well do state-of-the-art LLMs extract knowledge from semi-structured web data?

RQ2: Do the extracted knowledge triples, added upon original semi-structured data, improve answer accuracy?

RQ3: Can equipping LLMs with explicit knowledge extraction capabilities enhance their QA performance?

Our foremost contribution towards addressing these questions is an extension of the WebSRC

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2055–2074
March 24-29, 2026

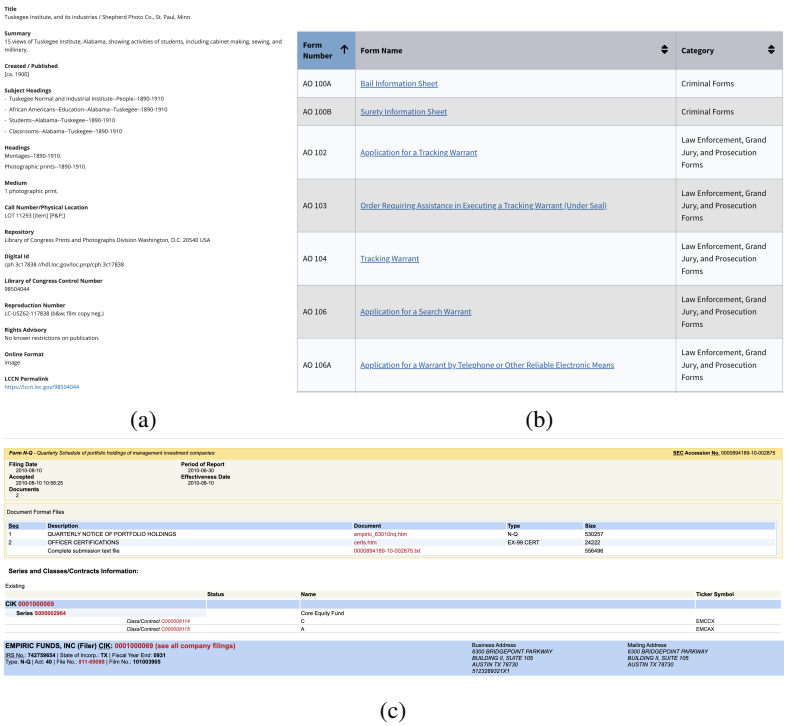

Figure 1: We categorize layouts of semi-structured data into three types: attribute-value pairs (e.g., Figure 1a), horizontal tables (e.g., Figure 1b), free-form (e.g., Figure 1c). Figure 1d lists the extracted triples from Figure 1b.

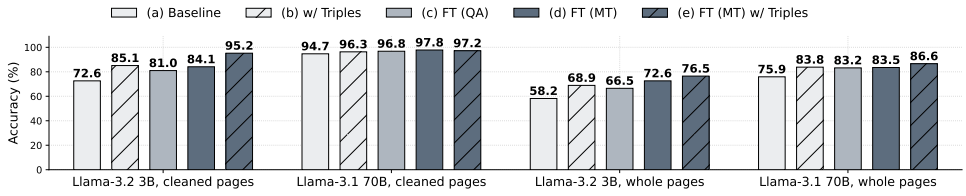

Figure 2: Enhancing semi-structured content with ground truth knowledge triples during inference (i.e., (b)) surpasses the baseline without augmentation (i.e., (a)). Multi-task fine-tuning with triple extraction (d) outperforms QA-only fine-tuning (i.e., (c)). Their combination (i.e., (e)) offers the greatest potential for improving QA performance in smaller models. (MT: multi-task, FT: fine-tuned.)

benchmark (Chen et al., 2021) with more realistic and challenging data[1]. WebSRC was originally designed for QA over cleaned semi-structured content. We enhance it by incorporating original whole webpages from additional websites, similar to those encountered in RAG tasks with web retrieval. Additionally, we introduce a knowledge extraction task by annotating ground-truth knowledge triples, to enable direct evaluation of extraction quality and facilitate analysis of how extracted knowledge impacts QA performance. (Section 2-3)

We conducted comprehensive experiments using state-of-the-art LLMs for knowledge extraction (**RQ1**). Our findings reveal that LLMs obtain F1 scores of 77% for triple extraction on cleaned semi-structured content, much higher than previous methods using GNN-based extraction (reported 46% F-measure (Lockard et al., 2020)). However, the score drops to 28% on whole webpages, showing that *web-scale knowledge extraction remains challenging even with LLMs*. (Section 4)

Our third contribution studies whether we can improve QA by augmenting the original semi-structured content with the extracted triples (**RQ2**). Our experiments show that although LLMs achieved high QA quality (95%) on cleaned semi-structured content, they *struggle on whole webpages as appearing in real RAG systems,* and the quality drops to 76%. Encouragingly, *augmenting the pages with ground-truth knowledge triples can increase QA accuracy* to 84%; despite the potential, we cannot take for granted the availability of

[1]We release the data for non-commercial use at https://github.com/facebookresearch/SemiBench.

ground truth triples in practice. (Section 5)

As a fourth contribution, we investigate whether improved capability of knowledge extraction can enhance LLMs' QA performance. We show that *multi-task fine-tuning on both QA and triple extraction tasks leads to improved QA performance even on unseen websites,* and the quality improvement is comparable to augmenting webpages with ground truth knowledge triples (83.5% vs. 83.8%). We also show that fine-tuning on triple extraction and providing extracted triples at inference time as augmentations offer *complementary benefits*, and adding them together can further improve QA quality to 87% (Figure 2). (Section 6).

Finally, as LLM-based knowledge extraction can be expensive, we investigate a new mechanism for knowledge extraction: we prompt LLMs with a few extraction examples, and let it write extraction scripts on semi-structured data. Remarkably, such methods can generalize across websites, yielding a 61% F1-score on cleaned semi-structured content, comparable to fine-tuned LLMs (51% (3B); 72% (70B)). Additionally, the extracted triples can bring a 5% gain in accuracy over zero-shot LLM-based QA—demonstrating *a promising pathway for scalable knowledge extraction*. Despite the potential, we have not seen success on whole webpages, where scripting is much harder.

Our studies suggested that *knowledge extraction can still effectively enhance LLMs' QA over semi-structured data*, through teaching of extraction capabilities and augmentation with extraction results. In addition to QA, knowledge extraction has broader values, such as offline indexing, knowledge integration, and user-facing presentation for improved interpretability (compared to unstructured content). Yet, *the current quality of extraction at web scale remains sub-optimal*, and LLM-based extraction is expensive especially at the web scale. Improving it, especially with the low-cost script-based approach, is essential for enabling high-stakes, large-scale applications.

2 Datasets

2.1 Problem definition

Semi-structured data is a type of data that has some level of organization, but does not conform to the rigid structure of traditional relational databases. There are three forms of semi-structured data. Each form is illustrated in Figure 1.

Attribute-value pairs / vertical tables (A-V): The data are in the form of attribute-value pairs. They may be in a table form, but with only two columns: one column for attributes, and one for values.

Horizontal web tables (Hz): The table typically has multiple rows and columns. Normally a row represents an entity and a column represents an attribute, and the corresponding cell gives the value. There are also cases where a column represents an entity and a row represents an attribute.

Free form semi-structured data (F-F): Free forms typically has a subject on the top of the page or a chunk, and key-value attributes in various layouts, horizontally aligned or vertically aligned; sometimes the predicates are skipped.

We consider the two problems below on semi-structured data.

Question answering. Given a question and an associated semi-structured reference document as input, the goal is to output the answer.

Triple extraction (TE). Given a semi-structured document as input, the goal is to generate a structured representation in text, typically in the form of (subject, predicate, object) triples.

2.2 The enriched WebSRC benchmark

Cleaned webpages. Part of our work is built upon WebSRC, the only large-scale semi-structured QA dataset available. Comprising question-answer pairs from 70 diverse websites across various topics and layouts, this dataset offers a comprehensive representation of the complexities inherent in semi-structured data. Webpages in this dataset have been cleaned, with noise elements such as ads and navigation tabs removed, and large objects truncated (Chen et al., 2021). Among these 70 websites, 60 have publicly accessible ground truth QA annotations. We focus on these 60 websites and extend them with a new task – triple extraction: we extract all (subject, predicate, object) triples from the HTML (HyperText Markup Language) of a webpage, in the order they appear on the page.

Specifically, we randomly selected three webpages from each of the 60 websites and manually annotated all triples present in these webpages. The subjects, predicates, and objects generally adhere to the original wording, primarily derived from text spans within the webpage. However, annotators were permitted to add supplementary disambiguation information when necessary; such information had to be enclosed in special characters. They may skip any incomplete triples (e.g., missing predi-

Metric	Value
Cleaned pages	
# QA pairs / # webpages	6,186 / 180
◇ A-V	1,089 / 72
◇ Hz	3,224 / 51
◇ F-F	1,873 / 57
# triples	4,741
# triples per page (average)	26.3
Whole pages	
# QA pairs / # webpages	1,006 / 251
# triples	30,263
# triples per page (average)	120.6

Table 1: Statistics of the enriched dataset.

cate) if a part is missing from the webpage. An example is shown in Figure 1b and 1d, and see Appendix A.4 for more details.

To facilitate the investigation of the connection between triple extraction and QA, we leveraged QA pairs from WebSRC that correspond to the same set of sampled webpages described above. Given the presence of paraphrased questions in WebSRC, we only retained one instance of each question with a unique ground truth answer.

Whole webpages. Additionally, we sampled 139 websites with semi-structured content from the top 2,500 registered domains ranked by page captures in Common Crawl[2]. For 56 of these sampled websites, we collected three webpages each; for the remaining 83 websites, we collected one webpage each. We manually annotated all triples on these webpages using the same process as for cleaned webpages, and developed human-audited QA pairs for each webpage with the support of LLMs. See Appendix A.5 for more details.

Table 1 summarizes the overall statistics.

2.3 Evaluation metrics

QA accuracy. We use $\textbf{Accuracy}_{\textbf{LM}}$, defined as the percentage of correctly answered questions, where we utilize an LLM (Llama 3.1-70B-Instruct) to check whether the response aligns with the ground truth. Llama was selected for its open-source nature and reproducibility. We verified human judgment against this metric for 150 random instances, with 99% agreement (149 out of 150). An additional rule-based metric is reported and discussed in Appendix A.2.

Triple extraction quality. We evaluate the performance of triple extraction using both global and triple-level matching metrics.

- Global match. We use **fuzzy match (FM)**, defined as the normalized character-level edit distance between the prediction and the ground truth, considering all extracted triples as a whole.

- Triple-level match. We first calculate the character-level edit distance between each pair of triples from the prediction and the ground truth, resulting in a score matrix. We then obtain the maximum weight matching of the score matrix using the Munkres algorithm (Munkres, 1957). Next, we employ an LLM (Llama 3.1-70B-Instruct) to check whether each pair of triple from the maximum weight matching are semantically the same. Based on this, we define the following metrics. $\textbf{P}_{\textbf{LM}}$ (LLM-based Precision): The percentage of extracted triples that are matched to ground-truth triples; $\textbf{R}_{\textbf{LM}}$ (LLM-based Recall): The percentage of ground-truth triples that are matched to extracted triples; $\textbf{F-1}_{\textbf{LM}}$ (LLM-based F-1 Score): The harmonic mean of precision and recall, calculated as $2 \cdot \mathrm{P_{LM}} \cdot \mathrm{R_{LM}} / (\mathrm{P_{LM}} + \mathrm{R_{LM}})$.

We manually verified the agreement between human judgment and the LLM for 150 randomly sampled triple pairs. The results showed a 95% agreement rate (142 out of 150), which we consider acceptable, given that some cases may be open to interpretation. We additionally report and discuss other triple-level metrics in Appendix A.2.

During evaluation, we disregard disambiguation information and incomplete triples in both the prediction and the ground truth.

3 Experimental Setup

3.1 Configurations

For cleaned webpages, we employ two distinct configurations when utilizing the datasets:

In-domain. We split the data such that we use the same set of websites during training and evaluation, with 120 webpages (two webpages per website) for training and 60 webpages (a webpage per website) for evaluation.

Out-of-domain. We split the data such that there is minimal domain overlap between the training and evaluation websites, with 90 webpages for training and 90 for evaluation.

For whole webpages, among the 56 websites from which we collected three webpages each, we used two webpages per website (a total of 112 webpages) for model training. The remaining webpages from these 56 websites were used for **in-domain** evaluation. Additionally, the 83 websites from which only one webpage was collected were used for **out-of-domain** evaluation.

[2] https://commoncrawl.org/.

Backbone	Setting	in-domain Global	in-domain Triple-level			out-of-domain Global	out-of-domain Triple-level		
		FM	P_{LM}	R_{LM}	$F\text{-}1_{LM}$	FM	P_{LM}	R_{LM}	$F\text{-}1_{LM}$
Llama 3.2-3B-Instruct	zero-shot	49.6	31.7	45.9	37.5	49.7	31.3	42.3	36.0
Llama 3.2-3B-Instruct	2-shot (in-domain) / 3-shot (out-of-domain)	48.7	50.4	73.6	59.8	45.7	26.5	29.2	27.8
Llama 3.2-3B-Instruct	fine-tuned	69.3	60.1	63.7	61.9	64.0	52.0	50.6	51.3
Llama 3.1-70B-Instruct	zero-shot	57.8	53.1	71.8	61.0	58.6	60.3	74.1	66.5
Llama 3.1-70B-Instruct	2-shot (in-domain) / 3-shot (out-of-domain)	90.2	88.4	92.0	90.2	74.1	69.4	70.1	69.7
Llama 3.1-70B-Instruct	fine-tuned	78.5	71.8	76.3	74.0	75.7	72.1	71.0	71.5
Claude 3.7 Sonnet	2-shot (in-domain) / 3-shot (out-of-domain)	92.4	89.9	92.8	91.3	78.3	76.0	77.4	76.7
GPT-4o	2-shot (in-domain) / 3-shot (out-of-domain)	94.1	93.9	95.5	94.7	75.9	76.4	76.9	76.6
generated scripts	single call	59.0	53.1	49.3	51.1	59.0	50.9	45.6	48.1
generated scripts	multiple calls with feedback	74.5	75.8	69.9	72.7	64.4	63.8	57.9	60.7

Table 2: Triple extraction performance on cleaned pages. All numbers are in percentage (%).

	Backbone	Page Cleaning	Setting	Global	Triple-level		
				FM	P_{LM}	R_{LM}	$F\text{-}1_{LM}$
in-domain	Llama 3.1-70B-Instruct	/	zero-shot	17.4	17.8	11.0	13.6
	Llama 3.1-70B-Instruct	Trafilatura	zero-shot	30.1	22.8	12.5	16.1
	Llama 3.1-70B-Instruct	Trafilatura	fine-tuned	34.4	22.8	15.1	18.2
out-of-domain	Llama 3.1-70B-Instruct	/	zero-shot	17.6	12.5	12.6	12.5
	Llama 3.1-70B-Instruct	Trafilatura	zero-shot	33.3	24.3	15.7	19.1
	Llama 3.1-70B-Instruct	Trafilatura	fine-tuned	39.5	27.1	21.4	23.9
	Claude 3.7 Sonnet	Trafilatura	zero-shot	32.7	26.2	23.0	24.5
	GPT-4o	Trafilatura	zero-shot	30.9	35.1	23.8	28.3

Table 3: Triple extraction performance on whole webpages. All numbers are in percentage (%).

3.2 Foundational models

We primarily focus on the experimental results obtained using Llama 3.2-3B-Instruct (L-3B) and Llama 3.1-70B-Instruct (L-70B) (Grattafiori et al., 2024). For a more comprehensive understanding, we also report the performance of Qwen 2.5-3B-Instruct (Q-3B), Qwen 2.5-72B-Instruct (Q-72B) (Qwen et al., 2025), GPT-4o (4o), GPT-4o mini (4o mini) (OpenAI et al., 2024), and Claude 3.7 Sonnet (3.7 Sonnet)[3] where relevant. Training and inference details are available in Appendix A.3, and prompt details are available in Appendix A.1.

4 Triple Extraction (RQ1)

4.1 Experiment setup

We considered three types of approaches for triple extraction:

(i) Off-the-shelf LLM-based extractors.
In-domain: We employ zero-shot and 2-shot extraction (Prompt 3 and 4 in Appendix A.1). For the 2-shot setting, each example contains a webpage from the same website, demonstrating triples that shall be extracted from the webpage.
Out-of-domain: We use zero-shot or 3-shot extraction, where the latter provides three examples each from the three forms of semi-structured content (i.e., attribute-value pair, horizontal table, and free-form). The out-of-domain setting illustrates web-scale extraction, where even providing two examples per website is still impractical.

(ii) Fine-tuned LLM-based extractors. We fine-tune the LLM using in-domain and out-of-domain training sets for respective configurations. Each training instance contains a webpage and the ground truth triples embraced by the page.

(iii) Triple extraction through automatically generated scripts. Running LLMs on each webpage can be prohibitively expensive. We employ Llama 3.1-70B-Instruct to generate a script for triple extraction from HTML for each website, and then run the scripts for triple extraction, which is much less resource-consuming.
- **Single call:** We provide the LLM with two exemplar instances of triple extraction, prompting it to generate a script. For in-domain extraction, the exemplar instances consist of HTML pages from the same website in the training set, along with their corresponding ground truth triples. For out-of-domain extraction, the exemplar instances consist of HTML pages from other webpages of the same website in the evaluation set, paired with triples extracted by the 3-shot extraction model, since ground truth triples are unavailable.
- **Multiple calls with feedback:** We generate multiple scripts through multiple calls and iteratively refine the generated scripts by executing the LLM-generated script and providing the execution results

[3] https://www.anthropic.com/news/claude-3-7-sonnet

	Augmentation	L-3B	L-70B	Q-3B	Q-72B	4o mini	4o
cleaned pages	no augmentation	72.6	94.7	81.6	95.1	93.3	96.3
	with ground truth triples	**85.1**	**96.3**	**92.4**	**97.1**	**97.0**	**96.7**
whole pages	no augmentation	58.2	75.9	61.0	81.4	77.1	86.3
	with ground truth triples	**68.9**	**83.8**	**72.3**	**85.1**	**84.1**	**90.5**

Table 4: Out-of-domain QA performance in $\text{Accuracy}_{\text{LM}}$ (%).

	Backbone	Cleaning	$\text{Accuracy}_{\text{LM}}$
in-domain	Llama 3.1-70B-Instruct	Trafilatura	49.3
	Llama 3.1-70B-Instruct	/	76.2
out-of-domain	Llama 3.1-70B-Instruct	Trafilatura	51.8
	Llama 3.1-70B-Instruct	/	75.9
	GPT-4o	/	86.3
	Claude 3.7 Sonnet	/	77.1

Table 5: Question answering performance on whole webpages in $\text{Accuracy}_{\text{LM}}$ (%).

as feedback to the LLM. We select the best one based on a chosen metric evaluated on the exemplar instances. See Appendix A.8 for more details.

We also discuss traditional triple extraction approaches in Appendix A.7.

4.2 Triple extraction quality

Cleaned pages. Table 2 shows various triple extraction approaches' performance on cleaned pages.

- In-domain extraction. We have the following observations. (1) Zero-shot triple extraction has mediocre performance on semi-structured data, obtaining triple-level F-1 of 61% with Llama 3.1-70B. (2) With only two examples from the same website, the 70B model quality (F-1) improved significantly to 90%, meaning as far as we manually annotate a couple of pages, we can achieve reasonable extraction quality. (3) The fine-tuned 70B model underperforms 2-shot, likely because the training data contains a broader set of 120 webpages from 60 websites, losing focus on the specific website during inference compared to 2-shot in-content learning. (4) Notably, LLM-based script-writing is almost comparable to the fine-tuned 70B model, with much lower computation resources for *inference*-time extractions. The inference time of the generated scripts on a single Xeon Platinum 8339HC core is $\sim$1/3000th of the inference time of a 3B LLM running on a single A100 (40GB) GPU. However, note that script coding poses considerable difficulties compared to directly understanding and converting semi-structured contents to triples; the multiple-call approach provides significant improvement over single-call for script generation (further ablation study in Appendix A.8).

- Out-of-domain extraction. In contrast, out-of-domain triple extraction is still challenging. We highlight three differences from in-domain extraction. First, compared to zero-shot, 3-shot in-context learning does not help much, since different websites often have very different formats. Second, fine-tuning helps better than 3-shot in-context learning; the improvement is marginal on 70B model (+2% on F1), but much more pronounced on 3B model (+24%). Finally, overall quality is much lower than in-domain—all state-of-the-art models have F1-score below 80%, showing that triple extraction is not ready for web scale even with LLMs.

- Triple extraction on different forms. Table 11 in Appendix summarizes the performance across the three different layouts. We observed the highest quality for the key-value form: even for out-of-domain, we achieved up to 90% F-1. In contrast, the quality for horizontal tables is lower (up to 81% F-1), and for free-form is the lowest (up to 63%). This is as expected, as there is the least commonality among websites for free-form data, making out-of-domain extraction more challenging.

Whole pages. Table 3 show the performance on whole webpages. Note that the context window of popular LLMs, including those we benchmarked, is typically limited to 128K tokens, which poses challenges for processing whole webpages. Specifically, the average and maximum token counts in our evaluation set (measured using the tokenizer of Llama 3.1) are 110K and 1.1M tokens, respectively. As a result, 29% of whole webpages exceed the 128K context window and cannot be processed. For these instances, we assign their quality metrics (i.e., P_{LM}, R_{LM}, etc.) a value of zero. To mitigate this issue, we apply Trafilatura (Barbaresi, 2021) for automatic page cleaning, which removes non-content noise (e.g., headers, footers, ads). This significantly reduces the average and maximum token counts to 6K and 105K, respectively. We did not benchmark few-shot extraction and LLM-based script writing performance because the maximum sequence length they require exceeds 128K tokens,

Additional reference	in-domain L-3B	in-domain L-70B	in-domain Q-3B	in-domain Q-72B	out-of-domain L-3B	out-of-domain L-70B	out-of-domain Q-3B	out-of-domain Q-72B
/	77.1	95.1	81.5	93.5	72.6	94.7	81.6	95.1
Script-extracted triples	80.6	94.9	87.5	94.2	77.7	92.9	86.5	94.2

Table 6: Zero-shot QA performance in $\text{Accuracy}_{\text{LM}}$ (%) on cleaned pages. Script-extracted triples improve QA quality for 3B models.

even after page cleaning. We highlight two observations: (1) The extraction quality is substantially lower, with the highest F-1_{LM} reaching only 28%, indicating a significant gap between triple extraction from cleaned versus whole pages. (2) Page cleaning generally contributes to improved triple extraction performance.

Summary: Even with the advanced understanding capabilities of LLMs, triple extraction remains a challenge on semi-structured content. Extraction on cleaned semi-structured content achieves up to 77% F1-score on out-of-domain websites; on full webpages, the F1-score drops to 28%, showing the difficulties in web-scale extraction. Encouragingly, we observe that script-based extraction achieves comparable results to few-shot extraction, offering an alternative low-cost medium-quality approach.

5 Improving QA Through Triple Augmentation (RQ2)

We next examine LLMs' quality in question answering leveraging semi-structured content, and whether or not augmenting the original content with the extracted knowledge triples can enhance QA performance. For this purpose, we conducted experiments where we concatenate the extracted triples obtained from a triple extraction model with the original reference text to create a new reference.

QA on semi-structured data. Table 4 reports results for out-of-domain QA. Whereas smaller models (i.e., Llama 3B, Qwen 3B, and GPT-4o mini) obtain only medium QA accuracy on cleaned pages, larger models (i.e., Llama 70B, Qwen 72B, and GPT-4o) already achieve high accuracy (94-96%). However, the quality drops significantly on whole webpages; even larger models achieve at best 86% QA accuracy. The nuance is that we use the flattened page for QA (see Appendix A.3 for details), which is shorter than the original HTML. Page cleaning using Trafilatura reduces the page length but at the cost of mistakenly missing a substantial amount of useful information (49.3% vs. 76.2%), highlighting the gap between information extraction from cleaned pages versus whole pages using LLMs (see Table 5).

Ground-truth triples. We start with ground truth triples for augmentation to illustrate the upper-bound benefit. Table 4 shows that incorporating ground truth triples can significantly enhance the QA performance: for smaller models, which have only medium quality on cleaned pages, the augmentation can improve by up to 13%; for larger models, which still fall short on whole pages, ground-truth triples bring substantial benefits and increase accuracy by up to 8%. Despite the potential benefits, ground truth triples are not readily available for augmentation.

Script-extracted triples. We further experiment with augmentation using script-extracted triples, which are more feasible in practice but may contain a level of noises. Table 6 presents overall QA quality. Even when augmented with script-extracted triples, we already observe much higher QA accuracy for 3B models for zero-shot QA, improving accuracy by up to 6%. Further analysis reveals that this trend also holds for 2-shot QA, and that the primary source of improvement comes from better performance on horizontal tables (Table 12 in Appendix). Larger models already obtain high QA accuracy on cleaned pages, thus do *not* gain additional benefits from script-based extractions because of the potential distractions caused by wrongly extracted triples.

Summary: Whereas LLMs already achieve high QA quality (96%) on cleaned semi-structured content, the quality drops by up to 20% on whole pages in the wild. Augmenting webpages with triples that capture factual knowledge from the semi-structured content can improve QA quality substantially. In practice, however, noisy triples such as those extracted by scripts, improve more for smaller LLMs but less for large LLMs that already achieve high QA quality.

6 Improving QA Through Enhanced Triple Extraction Capabilities (RQ3)

We next examine whether by teaching LLMs to obtain triple-extraction capability can enhance QA on

	Backbone	Setting	All	A-V	Hz	F-F
in-domain	L-3B	zero-shot	77.1	85.9	72.4	80.0
	L-3B	fine-tuned	98.2	97.0	99.6	96.5
out-of-domain	L-3B	zero-shot	72.6	81.1	70.1	81.8
	L-3B	fine-tuned	81.0	97.5	76.7	96.0
	L-70B	zero-shot	94.7	91.8	94.7	96.8
	L-70B	fine-tuned	96.8	99.1	96.7	95.6
	4o	zero-shot	96.3	94.0	96.3	98.0
	3.7 Sonnet	zero-shot	97.2	95.9	97.2	98.2

Table 7: Question answering performance in $\text{Accuracy}_{\text{LM}}$ (%) on cleaned pages.

FT task	L-3B	L-70B	Q-3B	Q-72B
/	72.6	94.7	81.6	95.1
QA	81.0	96.8	82.8	96.7
TE	74.3	94.6	81.4	95.3
QA + TE	**84.1**	**97.8**	**85.4**	**97.2**

Table 8: Out-of-domain QA performance in $\text{Accuracy}_{\text{LM}}$ (%) on cleaned pages.

FT task	Additional ref.	L-3B	L-70B	Q-3B	Q-72B
QA	/	81.0	96.8	82.8	96.7
	FT 3B ext. triples	78.6	/	82.7	/
	FT 70+B ext. triples	87.0	96.0	83.4	95.9
	Ground truth triples	94.6	96.9	93.5	96.8
QA + TE	/	84.1	97.8	85.4	97.2
	FT 3B ext. triples	84.0	/	84.1	/
	FT 70+B ext. triples	89.4	96.6	85.4	96.3
	Ground truth triples	95.2	97.2	94.5	97.1

Table 9: Out-of-domain QA performance in $\text{Accuracy}_{\text{LM}}$ (%) on cleaned pages.

FT task	Additional ref.	L-3B	L-70B	Q-3B	Q-72B
/	/	58.2	75.9	61.0	81.4
	Ground truth triples	68.9	83.8	72.3	85.1
QA	/	66.5	83.2	66.5	82.6
	FT 3B ext. triples	69.2	/	67.7	/
	FT 70+B ext. triples	70.1	82.3	68.6	82.3
	Ground truth triples	76.2	87.5	73.5	85.7
QA + TE	/	72.6	83.5	68.0	82.0
	FT 3B ext. triples	71.0	/	67.1	/
	FT 70+B ext. triples	70.1	84.5	71.6	82.3
	Ground truth triples	76.5	86.6	74.7	86.6

Table 10: QA performance on whole webpages from out-of-domain websites in $\text{Accuracy}_{\text{LM}}$ (%).

cleaned pages. We perform multi-task fine-tuning on both QA and triple extraction tasks. See Appendix A.3 for implementation details.

Cleaned pages. We summarize the QA performance of zero-shot and fine-tuned LLMs in Table 7. The fine-tuned Llama 3B achieves strong performance (98%) on in-domain QA.

We have three observations on out-of-domain QA. (1) It is harder than in-domain. Top commercial models (GPT-4o, Claude 3.7 Sonnet) and Llama 70B all demonstrate strong zero-shot performance (up to 97% in accuracy) in question answering. (2) Fine-tuning lifts QA accuracy, significantly for the 3B model and marginally for the 70B model. (3) Fine-tuning normally helps when original quality is sub-optimal, such as on attribute-value pairs for the 70B model, and on free-form for the 3B model; however, even after fine-tuning, a major gap remains on horizontal tables for 3B model.

- Effect of multi-task training. We next investigate if the knowledge-extraction capability improves QA capability. Table 8 compares the performance of LLMs fine-tuned on different tasks. Multi-task fine-tuning with triple extraction provides additional benefits, particularly for 3B models. Moreover, as further detailed in Table 17 in the Appendix, multi-task fine-tuning primarily and consistently improves QA accuracy on the two harder forms: horizontal tables and free-form data, over sole fine-tuning on the QA task.

- Adding in addition triple augmentation. Finally, we augment the input with extracted triples and run the fine-tuned models. Table 9 reports results for out-of-domain QA. We observe that fine-tuning is less effective than triple augmentation. For example, multi-task fine-tuning of Llama 3B for QA improves quality by 3%, while adding triples from 70+B models boosts it by 6%. With that being said, these two approaches are complementary to each other. Continuing with the same example, combining the two improves QA accuracy by 8%. Despite the combined power, we point out that the QA models generally do not benefit from triples extracted by fine-tuned models of the same size, which do not seem to add additional value.

Whole pages. The conclusions drawn from cleaned pages generally hold for whole pages, as shown in Table 10. Since larger models have only medium QA quality on whole webpages, we observed much higher quality gain with fine-tuning (up to 8%), and multi-task fine-tuning improves further (6%) over fine tuning with only QA tasks. On the other hand, incorporating triples extracted by 70+B models may degrade performance, likely due to their lower triple extraction accuracy on whole pages.

Summary: We show that we can further improve QA accuracy by fine-tuning LLMs with triple extraction tasks, and the improvement is comparable to using triple augmentation. When combined with triple augmentation, we observe a quality boost (+11% with ground truth triples on full webpages).

7 Related Work

Benchmarks. Several publicly available benchmarks exist for semi-structured websites, including SWDE (Hao et al., 2011) and WEIR (Bronzi et al., 2013), which focus on extracting predefined attributes of entities. In contrast, Expanded SWDE (Lockard et al., 2019) aims to recover all (subject, predicate, object) triples for *detail webpages* that each contains information about a particular entity. Additionally, WebSRC (Chen et al., 2021) is a dataset aimed at answering questions about semi-structured webpages. Our dataset distinguishes itself by covering both knowledge triple extraction and question answering tasks.

Approaches. Early works have explored directly providing an answer to user questions by knowledge extraction from webpages (Mausam, 2016), from distantly-supervised knowledge extraction (Dong et al., 2014; Lockard et al., 2018) to GNN-based zero-shot knowledge extraction (Lockard et al., 2020). The success was capped by our capability of precisely extracting knowledge and leveraging the extracted knowledge which can contain errors. In contrast, we have observed more success in leveraging web tables (Cafarella et al., 2018), which takes the retrieval approach to return a webtable to answer the user question, bypassesing the hard retrieval and summarization steps. In recent years, there has been significant progress in knowledge extraction and QA with the help of pre-trained language models (Xie et al., 2021; Zhao et al., 2022; Deng et al., 2022; Wang et al., 2022; Zhang et al., 2024; Hong et al., 2024) and LLMs (Sarkhel et al., 2023; Tan et al., 2025). Following this trend and aiming to shed insights on this evolving landscape of web-based QA, we are the first to assess whether knowledge triple extraction can still provide value for QA over semi-structured content in the era of LLMs.

Another line of research has focused on leveraging accessible structured data sources, such as KGs, tables, or databases, for QA (Jiang et al., 2023), and employing direct multimodal LLMs that process semi-structured data presented as images (Faysse et al., 2025; He et al., 2024). In contrast, our work is motivated by the standard web-based RAG setting, where the semi-structured data arises from retrieval results without relying on additional dedicated, cleaned artifacts like databases, nor framing the problem as image or multimodal understanding.

8 Conclusion

We show that while LLMs excel at QA over semi-structured web data, knowledge extraction still offers measurable benefits, especially for smaller models. However, the incremental value diminishes for top LLMs on cleaned pages, highlighting the need for further research on real-world, noisier whole pages. Improving extraction quality remains essential for broader applications beyond QA, such as knowledge integration and interpretability.

Finally, we summarize the practical takeaways and recommendations as follows:

Triple extraction.

- In general, large models (70+B) achieve reasonable triple extraction quality with prompting alone. When webpages exceed the context window, preprocessing (e.g., cleaning with tools such as Trafilatura) can be beneficial. By contrast, small models (3B) are inadequate for high-quality extraction.
- LLM-generated extraction scripts provide a cost-effective alternative with comparable quality, especially for websites with a large number of similarly structured pages.

QA on semi-structured webpages.

- Fine-tuned 3B models perform well on in-domain, cleaned pages; for out-of-domain or uncleaned pages, larger models are typically required.
- Multi-task fine-tuning on QA and knowledge extraction can improve QA accuracy.
- Concatenating the original webpages with (potentially noisy) extracted triples can further improve QA performance, even for fine-tuned models.

Limitations

Script Writing for Whole Pages. Our LLM-based script writing approach was not further discussed for whole web pages due to context length constraints. Beyond applying Trafilatura for automatic content cleaning, we did not extensively experiment with alternative strategies for reducing token count. This decision reflects the primary focus of our work: to provide a comprehensive evaluation of the current capability boundaries of LLMs, rather than to propose or optimize solutions for this specific challenge. We acknowledge that developing

more effective approaches for handling long and noisy web pages is an important direction for future research, and we encourage further exploration in this area.

Nevertheless, in Appendix A.6, we report supplementary results with Llama 4 (Adcock et al., 2026), which has a larger context length. These results demonstrate that the findings for script writing on whole pages are consistent with those observed in the cleaned page setting. Furthermore, we refer readers to our follow-up work (Liu et al., 2025), which addresses the aforementioned challenges in script writing for whole pages.

Acknowledgments

We would like to thank the anonymous area chair and reviewers for their careful reviews and feedback. We also thank Shicheng Liu from Stanford University for his valuable comments and helpful discussions.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Aaron Adcock, Aayushi Srivastava, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pande, Abhinav Pandey, Abhinav Sharma, Abhishek Kadian, Abhishek Kumawat, Adam Kelsey, and 1 others. 2026. The llama 4 herd: Architecture, training, evaluation, and deployment notes. *arXiv preprint arXiv:2601.11659*.

Adrien Barbaresi. 2021. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131, Online. Association for Computational Linguistics.

Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. 2013. Extraction and integration of partially overlapping web sources. *Proc. VLDB Endow.*, 6(10):805–816.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. Rebel: Relation extraction by end-to-end language generation. In *Findings of the association for computational linguistics: emnlp 2021*, pages 2370–2381.

Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten years of webtables. *Proc. VLDB Endow.*, 11(12):2140–2149.

Xingyu Chen, Zihan Zhao, Lu Chen, JiaBao Ji, Danyang Zhang, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. WebSRC: A dataset for web-based structural reading comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4173–4185, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michael Han Daniel Han and Unsloth team. 2023. Unsloth.

Xiang Deng, Prashant Shiralkar, Colin Lockard, Binxuan Huang, and Huan Sun. 2022. Dom-lm: Learning generalizable representations for html documents. *Preprint*, arXiv:2201.10608.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 601–610, New York, NY, USA. Association for Computing Machinery.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1535–1545.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. *Preprint*, arXiv:2407.01449.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

P. Gulhane, A. Madaan, R. R. Mehta, J. Ramamirtham, R. Rastogi, S. Satpal, S. H. Sengamedu, A. Tengli, and C. Tiwari. 2011. Web-scale information extraction with vertex. In *ICDE*.

Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR

'11, page 775–784, New York, NY, USA. Association for Computing Machinery.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand. Association for Computational Linguistics.

Zhi Hong, Kyle Chard, and Ian Foster. 2024. Combining language and graph models for semi-structured information extraction on the web. *Preprint*, arXiv:2402.14129.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 311–316, Vancouver, Canada. Association for Computational Linguistics.

N. Kushmerick, D. S. Weld, and R. B. Doorenbos. 1997. Wrapper induction for information extraction. In *IJCAI*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Shicheng Liu, Kai Sun, Lisheng Fu, Xilun Chen, Xinyuan Zhang, Zhaojiang Lin, Rulin Shao, Yue Liu, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. 2025. Scribes: Web-scale script-based semi-structured data extraction with reinforcement learning. *Preprint*, arXiv:2510.01832.

Colin Lockard, Xin Luna Dong, Arash Einolghozati, and Prashant Shiralkar. 2018. Ceres: distantly supervised relation extraction from the semi-structured web. *Proc. VLDB Endow.*, 11(10):1084–1096.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. OpenCeres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3047–3056, Minneapolis, Minnesota. Association for Computational Linguistics.

Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. 2020. ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8105–8117, Online. Association for Computational Linguistics.

Mausam Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 4074–4077. AAAI Press.

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Ritesh Sarkhel, Binxuan Huang, Colin Lockard, and Prashant Shiralkar. 2023. Self-training for label-efficient information extraction from semi-structured web-pages. *Proc. VLDB Endow.*, 16(11):3098–3110.

Rohini Srihari, Wei Li, and X Li. 1999. Information extraction supported question answering. In *TREC*.

Jiejun Tan, Zhicheng Dou, Wen Wang, Mang Wang, Weipeng Chen, and Ji-Rong Wen. 2025. Htmlrag: Html is better than plain text for modeling retrieved knowledge in rag systems. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 1733–1746. ACM.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. 2022. Webformer: The web-page transformer for structure information extraction. *Preprint*, arXiv:2202.00217.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz,

Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chenhao Xie, Wenhao Huang, Jiaqing Liang, Chengsong Huang, and Yanghua Xiao. 2021. Webke: Knowledge extraction from semi-structured web with pre-trained markup language model. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 2211–2220, New York, NY, USA. Association for Computing Machinery.

Ge Zhang, Jian Wang, and Jing Ling Wang. 2024. Joint information extraction from semi-structured web pages using BERT. In *Fourth International Conference on Advanced Algorithms and Neural Networks (AANN 2024)*, volume 13416, page 134161X. International Society for Optics and Photonics, SPIE.

Zihan Zhao, Lu Chen, Ruisheng Cao, Hongshen Xu, Xingyu Chen, and Kai Yu. 2022. TIE: Topological information enhanced structural reading comprehension on web pages. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1808–1821, Seattle, United States. Association for Computational Linguistics.

Xiaohan Zou. 2020. A survey on application of knowledge graph. *Journal of Physics: Conference Series*, 1487(1):012016.

A Appendix

A.1 List of prompts

Prompt 1: Prompt employed in computing LLM-based QA metrics.

<system>
You need to check whether the prediction of a question-answering system to a question is correct. You should make the judgment based on the ground truth answer provided to you. Your response should be "correct" if the prediction is correct or ïncorrectïf the prediction is wrong.
<user>
Question: {question}
Ground truth: {ground_truth}
Prediction: {prediction}
Correctness:

Prompt 2: Prompt employed in computing LLM-based triple extraction metrics.

<system>
You are given two (subject, predicate, object) triples. Your response should be "Yes" if the triples are semantically the same or "No" if they are semantically different.
<user>
{triple_1}
{triple_2}

Prompt 3: Triple extraction prompt.

<system>
You are given a doc in HTML and its title. Please return all (subject, predicate, object) triples that can be extracted from the doc, in the order they appear in the doc. Subject, predicate, and object should generally be gained from the text spans in the doc or the title. Please only include complete triples; if for any section the predicate or object is missing from the doc, you may skip it. Each line in your response should be a triple.
<user>
title
{title}
HTML
{html}

Prompt 4: Triple extraction prompt with demonstrations.

<system>
You are given a doc in HTML and its title. Please return all (subject, predicate, object) triples that can be extracted from the doc, in the order they appear in the doc. Subject, predicate, and object should generally be gained from the text spans in the doc or the title. Please only include complete triples; if for any section the predicate or object is missing from the doc, you may skip it. Each line in your response should be a triple.
Example 1
Input
title:
{title_1}
HTML:
{html_1}
Output
{triples_1}
Example 2
Input
title:
{title_2}
HTML:
{html_2}
Output
{triples_2}
<user>
title
{title}
HTML
{html}

Prompt 5: Prompt employed by *GenerateScriptFromLLM* in Algorithm 1 #4.

<system>
Your task is to write a program following the instruction. Your response should be a Python function(html: str) only without extra words.
<user>
Please write a Python function parse(html: str) to extract all (subject, predicate, object) triples from the html.
The return value should be a list of triples, in the order they appear in the html.
Subject, predicate, and object should generally be gained from the text spans in the doc.
The return value should only include complete triples; if for any section the predicate or object is missing from the doc, it may be skipped.
Below is an sample of Input and Output
Input (html)
<head><title>{title}</title></head>
{html}
Output (triples)
{triples}

Prompt 6: Prompt employed by *GenerateScriptFromLLM* in Algorithm 1 #2.

<system>
Your task is to write a program following the instruction. Your response should be a Python function(html: str) only without extra words.
<user>
Please write a Python function parse(html: str) to extract all (subject, predicate, object) triples from the html.
The return value should be a list of triples, in the order they appear in the html.
Subject, predicate, and object should generally be gained from the text spans in the doc.
The return value should only include complete triples; if for any section the predicate or object is missing from the doc, it may be skipped.
Below are two samples of Input and Output
Sample 1
Input (html)
<head><title>{title_1}</title></head>
{html_1}
Output (triples)
{triples_1}
Sample 2
Input (html)
<head><title>{title_2}</title></head>
{html_2}
Output (triples)
{triples_2}

Prompt 7: Prompt employed by *GenerateScriptFromLLM* in Algorithm 1 #8.

<system>
Your task is to fix/improve a program following the instruction if possible. Your response should be a Python function(html: str) only without extra words.
<user>
Please fix/improve a Python function parse(html: str) to extract all (subject, predicate, object) triples from the html.
The return value should be a list of triples, in the order they appear in the html.
Subject, predicate, and object should generally be gained from the text spans in the doc.
The return value should only include complete triples; if for any section the predicate or object is missing from the doc, it may be skipped.
Below is an sample of Input and Output
Input (html)
<head><title>{title}</title></head>
{html}
Output (triples)
{triples}
Here is the function and its execution result given the sample input:
Function
{previous_script}
Execution result
{execution_result}

Prompt 8: Prompt for QA with reference.

<system>
You are given a question and a reference that may or may not help answer the question. Please answer the question. Be concise.
<user>
Question
{question}
Reference
{reference}

Prompt 9: Prompt for QA without reference.

<system>
Please answer the question. Be concise.
<user>
Question
{question}

Prompt 10: Prompt for QA with demonstrations.

<system>
You are given a question and a reference that may or may not help answer the question. Please answer the question. Be concise.
Example 1
Question
{question_1}
Reference
{reference_1}
Output
{answer_1}
Example 2
Question
{question_2}
Reference
{reference_2}
Output
{answer_2}
<user>
Question
{question}
Reference
{reference}

	Backbone	Setting	Global FM			Triple-level F-1_{LM}		
			A-V	Hz	F-F	A-V	Hz	F-F
in-domain	Llama 3.2-3B-Instruct	zero-shot	54.5	51.8	41.4	42.5	39.6	27.6
	Llama 3.2-3B-Instruct	2-shot	54.0	44.4	45.8	62.9	66.1	49.7
	Llama 3.2-3B-Instruct	fine-tuned	82.4	60.9	60.3	71.7	53.0	57.1
	Llama 3.1-70B-Instruct	zero-shot	62.6	65.6	44.8	64.3	76.0	40.4
	Llama 3.1-70B-Instruct	2-shot	95.3	84.9	88.6	92.4	85.3	91.4
	Llama 3.1-70B-Instruct	fine-tuned	89.3	81.8	61.9	83.2	80.1	56.5
	Claude 3.7 Sonnet	2-shot	96.0	92.8	87.6	93.4	91.6	88.1
	GPT-4o	2-shot	97.5	91.4	92.1	96.5	92.2	94.6
	generated scripts	single call	67.3	64.7	43.3	55.7	57.4	39.7
	generated scripts	multiple calls with feedback	84.8	75.9	60.3	85.1	73.2	56.0
out-of-domain	Llama 3.2-3B-Instruct	zero-shot	66.1	42.5	43.0	54.8	24.3	32.1
	Llama 3.2-3B-Instruct	3-shot	62.3	40.0	35.8	53.1	13.4	21.3
	Llama 3.2-3B-Instruct	fine-tuned	84.4	56.3	53.0	74.0	35.0	54.2
	Llama 3.1-70B-Instruct	zero-shot	66.7	60.2	45.0	71.1	68.9	51.1
	Llama 3.1-70B-Instruct	3-shot	86.0	69.2	68.8	83.5	67.5	56.1
	Llama 3.1-70B-Instruct	fine-tuned	85.8	73.8	66.5	76.2	72.7	63.2
	Claude 3.7 Sonnet	3-shot	87.2	79.2	65.0	87.0	80.6	53.9
	GPT-4o	3-shot	87.6	75.4	61.8	89.8	76.6	58.0
	generated scripts	single call	78.4	54.5	43.2	64.5	46.4	30.1
	generated scripts	multiple calls with feedback	82.7	57.3	55.2	81.0	56.2	43.5

Table 11: Global FM and triple-level F-1_{LM} across different layouts on cleaned pages. All numbers are in percentage (%).

Setting	Additional reference	L-3B				Q-3B			
		All	A-V	Hz	F-F	All	A-V	Hz	F-F
zero-shot	/	77.1	85.9	72.4	80.0	81.5	88.1	79.9	80.4
	Script-extracted triples	80.6	87.6	79.2	78.9	87.5	89.2	87.8	85.9
	Ground truth triples	89.1	91.2	88.8	88.5	89.2	89.8	91.4	85.1
2-shot	/	75.6	91.2	69.3	77.5	83.1	95.0	78.7	83.7
	Script-extracted triples	82.9	89.8	83.5	77.8	86.8	95.0	87.1	81.6
	Ground truth triples	85.6	88.7	85.4	84.0	89.4	94.8	90.3	84.7

Table 12: In-domain QA performance in Accuracy_{LM} (%) across different layouts on cleaned pages. Script-extracted triples enhance QA results with the most pronounced improvement observed in horizontal layout.

Setting	Additional reference	L-3B	L-70B	Q-3B	Q-72B
zero-shot	/	63.9	81.7	63.9	81.4
	Script-extracted triples	70.6	84.6	72.1	82.9
2-shot	/	69.8	92.3	79.6	93.0
	Script-extracted triples	75.3	92.7	81.5	94.2

Table 13: In-domain QA performance in Accuracy_{A} (%) on cleaned pages.

FT task	L-3B	L-70B	Q-3B	Q-72B
/	65.2	87.8	67.7	88.4
QA	79.9	95.9	80.1	93.7
TE	68.5	88.1	63.0	89.5
QA + TE	84.4	97.2	82.7	94.2

Table 14: Out-of-domain QA performance in Accuracy_{A} (%) on cleaned pages.

Fine-tuning task	Additional reference	L-3B	L-70B	Q-3B	Q-72B
QA	/	79.9	95.9	80.1	93.7
	FT 3B extracted triples	77.3	/	79.8	/
	FT 70+B extracted triples	85.0	94.2	82.0	94.4
	Ground truth triples	93.2	94.8	89.9	93.1
QA + Triple Extraction	/	84.4	97.2	82.7	94.2
	FT 3B extracted triples	83.2	/	80.8	/
	FT 70+B extracted triples	87.5	95.4	83.7	94.9
	Ground truth triples	94.0	96.2	90.8	93.6

Table 15: Out-of-domain QA performance in Accuracy_{A} (%) on cleaned pages.

	Backbone	Setting	Triple-level						
			EM	P_{EM}	R_{EM}	$F\text{-}1_{EM}$	P_{FM}	R_{FM}	$F\text{-}1_{FM}$
in-domain	Llama 3.2-3B-Instruct	zero-shot	0.4	0.4	2.0	0.6	41.4	65.3	50.7
	Llama 3.2-3B-Instruct	2-shot	42.4	42.4	61.2	50.1	60.2	87.8	71.4
	Llama 3.2-3B-Instruct	fine-tuned	40.7	47.6	45.5	46.5	74.6	77.9	76.2
	Llama 3.1-70B-Instruct	zero-shot	8.0	8.5	12.3	10.1	54.3	76.0	63.4
	Llama 3.1-70B-Instruct	2-shot	75.3	77.7	81.7	79.6	90.3	92.8	91.5
	Llama 3.1-70B-Instruct	fine-tuned	50.6	52.7	54.3	53.5	78.6	82.7	80.6
	Claude 3.7 Sonnet	2-shot	79.8	80.5	83.4	81.9	92.9	94.9	93.9
	GPT-4o	2-shot	81.8	83.9	83.0	83.5	93.8	94.1	94.0
	generated scripts	single call	30.9	58.8	32.0	41.5	57.5	52.7	55.0
	generated scripts	multiple calls with feedback	56.1	73.6	56.9	64.2	81.1	75.6	78.2
out-of-domain	Llama 3.2-3B-Instruct	zero-shot	4.5	4.5	6.6	5.3	44.2	58.8	50.5
	Llama 3.2-3B-Instruct	3-shot	9.3	15.5	10.3	12.4	37.5	35.7	36.6
	Llama 3.2-3B-Instruct	fine-tuned	26.7	29.4	28.9	29.2	70.8	66.3	68.5
	Llama 3.1-70B-Instruct	zero-shot	8.8	9.4	11.4	10.3	60.3	73.5	66.3
	Llama 3.1-70B-Instruct	3-shot	42.1	45.3	46.2	45.7	76.5	76.2	76.4
	Llama 3.1-70B-Instruct	fine-tuned	41.1	45.3	42.9	44.1	79.1	77.0	78.0
	Claude 3.7 Sonnet	3-shot	48.1	49.6	51.2	50.4	80.8	83.2	82.0
	GPT-4o	3-shot	40.1	41.7	42.5	42.1	78.1	78.5	78.3

Table 16: Triple extraction performance on cleaned pages. All numbers are in percentage (%).

Fine-tuning task	L-3B			L-70B			Q-3B			Q-72B		
	A-V	Hz	F-F	A-V	Hz	F-F	A-V	Hz	F-F	A-V	Hz	F-F
QA	97.5	76.7	96.0	99.1	96.7	95.6	94.7	81.5	82.8	**99.1**	96.5	96.0
QA + Triple Extraction	**97.8**	**80.3**	**97.8**	**99.4**	**97.5**	**98.6**	**97.2**	**84.4**	**84.2**	96.5	**97.0**	**98.6**

Table 17: Out-of-domain QA performance in $\text{Accuracy}_{\text{LM}}$ (%) across different layouts on cleaned pages. Compared to solely fine-tuning on the QA task, multi-task fine-tuning with triple extraction consistently improves results on horizontal (Hz) tables and free-form (F-F) data.

Reference	$\text{Accuracy}_{\text{A}}$		$\text{Accuracy}_{\text{LM}}$	
	L-3B	L-70B	L-3B	L-70B
(None)	3.6	7.4	4.4	11.5
HTML	62.8	88.0	69.2	94.0
Flattened	65.2	87.8	72.6	94.7

Table 18: Zero-shot QA accuracy (%) on cleaned pages.

	Global	Triple-level		
	FM	P_{LM}	R_{LM}	$F\text{-}1_{LM}$
Llama 4-Maverick-17B-128E, zero-shot	31.5	20.0	10.1	13.4
generated scripts, single call	23.5	15.7	11.0	12.9
generated scripts, multiple calls with feedback	27.9	19.0	14.5	16.4

Table 19: Triple extraction performance on whole webpages utilizing Llama 4 Maverick. All numbers are in percentage (%).

A.2 Discussion of alternative metrics

Compared to LLM-based metrics, rule-based metrics offer the advantages of being more deterministic and having lower computational costs. Therefore, we define and report additional rule-based metrics for both QA and triple extraction tasks.

QA accuracy. We report **Appearance-based Accuracy (Accuracy$_{\text{A}}$)**, where the correctness is determined by whether the ground truth answer appears within the first 50 tokens of the model's response after normalization (i.e., lowercase, space normalization, remove leading and ending punctuations).

Table 14 and Table 15 report the QA performance in Accuracy$_{\text{A}}$, which correspond to Table 8 and Table 9, respectively. Overall, we can observe that the performance measured in Accuracy$_{\text{LM}}$ is generally higher than Accuracy$_{\text{A}}$, especially for the zero-shot setting. This is because LLMs, particularly when not fine-tuned, occasionally provide a correct response that does not strictly contain the ground truth answer as a substring. This suggests that the real performance can be underestimated by Accuracy$_{\text{A}}$. Nevertheless, Accuracy$_{\text{A}}$ aligns with Accuracy$_{\text{LM}}$ in terms of the relative performance and conclusions.

Triple extraction quality. Let A be the set of triples in the prediction and B be the set of triples in the ground truth, we define exact match-based metrics as follows, where we consider two triples are the same if subject, predicate, and object are the same after normalization (i.e., lower-casing, space normalization, and punctuation stripping).

- **Exact Match (EM)**:= $|A \cap B| / \max(|A|, |B|)$
- **EM-based Precision (P_{EM})**:= $|A \cap B| / |A|$
- **EM-based Recall (R_{EM})**:= $|A \cap B| / |B|$
- **EM-based F-1 Score (F-1_{EM})**:= $2 \cdot \text{P}_{\text{EM}} \cdot \text{R}_{\text{EM}} / (\text{P}_{\text{EM}} + \text{R}_{\text{EM}})$

Additionally, we define **Fuzzy Match-based Precision P_{FM}, Recall (R_{FM})**, and **F-1 Score (F-1_{FM})**. These metrics are analogous to their LLM-based counterparts, with the exception that we utilize the mean character-level fuzzy match scores of the triple pairs from the maximum weight matching as the numerator in calculating precision and recall, rather than relying on the matching rate verified by the LLM.

Table 16 shows the triple extraction performance in the exact match-based and fuzzy match-based metrics, which corresponds to Table 2.

A.3 Implementation details

Training and inference. When the target task is triple extraction, we pass the webpage in HTML to LLMs as input. For question answering and multi-task training, we use the flattened page for page representation obtained by Beautiful Soup[4], as our preliminary study (Table 18) showed that it yields slightly better performance compared to using HTML directly. Additionally, our preliminary study (Table 18) validated that the answers in our experiments are mainly generated through RAG, as LLMs without references (Prompt 9) achieve an accuracy lower than 12%.

We generally used Prompt 3 and Prompt 4 for triple extraction, and Prompt 8, Prompt 9, and Prompt 10 for question answering. These prompts were employed with nuanced adaptations as needed (e.g., add an additional demonstration according to the particular experimental setting). In particular, when using the flattened page as a reference, the reference was set to be the concatenation of the title (provided separately in the WebSRC dataset) and the flattened text (returned by Beautiful Soup after passing the HTML to it).

We leveraged Transformers (Wolf et al., 2020) and Unsloth (Daniel Han and team, 2023) for supervised fine-tuning on A100 (40G) or H100 (80GB) GPUs. In all fine-tuning experiments, we employed QLoRA (Dettmers et al., 2023) with settings r=16 and alpha=32, and utilized prompt-masking. Fine-tuning was performed for 1 epoch, with a batch size of 2, and learning rates of 2e-4 for 3B models and 5e-5 for 70+B models.

Evaluation. Our LLM-based metrics involve a two-stage process. First, we apply straightforward rules to assess clear-cut cases (i.e., the triples match exactly, the response matches perfectly the ground truth answer). For the remaining cases, we rely on LLMs, employing Prompt 1 for QA metrics and Prompt 2 for triple extraction metrics.

A.4 Additional annotation examples and details

For the example in Figure 1a, the extracted triples include:

[4] https://www.crummy.com/software/BeautifulSoup/.

- ("Tuskegee Institute, and its industries / Shepherd Photo Co., St. Paul, Minn.", "Title", "Tuskegee Institute, and its industries / Shepherd Photo Co., St. Paul, Minn.")
- ("Tuskegee Institute, and its industries / Shepherd Photo Co., St. Paul, Minn.", "Summary", "15 views of Tuskegee Institute, Alabama, showing activities of students, including cabinet making, sewing, and millinery.")
- ("Tuskegee Institute, and its industries / Shepherd Photo Co., St. Paul, Minn.", "Created / Published", "[ca. 1900]")

and so on. The subject is "Tuskegee Institute, and its industries / Shepherd Photo Co., St. Paul, Minn." for all triples in this example. The predicates come from texts in bold, and the texts following each predicate are the objects.

For the example in Figure 1c, the extracted triples include:

- ("EDGAR Filing Documents for 0000894189-10-002875", "Filing Date", "2010-08-10")
- ("empiric_63010nq.htm", "Description", "QUARTERLY NOTICE OF PORTFOLIO HOLDINGS")
- ("Series S000002964", "Name", "Core Equity Fund")

and so on. Note that "EDGAR Filing Documents for 0000894189-10-002875" comes from the webpage title (not in the screenshot).

For the webpage shown in Figure 3, predicates such as "CPU" and "GPU" do not have corresponding objects. As a result, they do not form complete triplets and can be skipped. However, there are two instances of the predicate "Base" with different meanings. In such cases, we may add disambiguation information to the predicates to clarify their context.

We conducted two phases of annotation validation:

- **Phase 1:** After the initial annotation (performed by a group of five annotators with a linguistic background), 20% of the annotations were randomly sampled and carefully checked by the paper's authors. The authors then provided feedback to the annotation group on common issues observed during this review.
- **Phase 2:** Based on this feedback, the most experienced annotator in the group manually reviewed the quality of all annotations and corrected any issues as necessary.

A.5 Details of QA pair construction for whole webpages

To prevent generating questions solely about content that the LLMs easily interpret, an initial set of candidate QA pairs was generated by prompting a 70B Llama model with the webpage content and *corresponding ground truth triples*. We then refined the set with the LLMs by: (1) Removing questions that require heavy reasoning, as we focus on evaluating the ability to comprehend semi-structured webpages rather than heavily reasoning over them. (2) Removing questions that combined multiple distinct questions into one (e.g., "What is an atom and how big is it?") to avoid artificially increasing difficulty by combining multiple questions. (3) Filtering out questions that were correctly answered by all LLMs from the set {Llama 3.1-70B-Instruct, Llama 3.1-8B-Instruct, GPT-4o, GPT-4o mini, Claude 3.7 Sonnet, Gemma 3-27b-it (Team et al., 2025), Phi-3.5-MoE-instruct (Abdin et al., 2024)} to create questions that effectively differentiate model performance. Finally, we performed human audits to eliminate QA pairs where questions were not grounded in the webpage content or where answers were incorrect.

A.6 Script writing for whole pages

We utilize Llama 4 Maverick (Adcock et al., 2026) (with a 1M context window) and compare direct LLM extraction to script-based extraction in Table 19. We report performance only for webpages in the in-domain evaluation set, as only a single webpage was collected for each website in the out-of-domain evaluation set in our dataset. Trafilatura was applied for all runs. Our findings, which are consistent with those observed in the cleaned page setting, are as follows: (1) The multiple-call approach outperforms the single-call approach. (2) The quality of triples extracted by generated scripts is comparable to that of direct LLM extraction.

		P_{LM}	R_{LM}	F-1_{LM}
cleaned pages, in-domain	ReVerb	0.7	0.5	0.6
	REBEL-large	8.5	2.1	3.4
	Llama 3.1-70B-Instruct, zero-shot	53.1	71.8	61.0
	Llama 3.1 generated scripts, multiple calls with feedback	75.8	69.9	72.7
cleaned pages, out-of-domain	ReVerb	0.0	0.0	0.0
	REBEL-large	8.3	1.9	3.1
	Llama 3.1-70B-Instruct, zero-shot	60.3	74.1	66.5
	Llama 3.1 generated scripts, multiple calls with feedback	63.8	57.9	60.7
whole pages, in-domain	ReVerb	2.3	1.2	1.5
	REBEL-large	1.1	1.2	1.1
	Llama 4-Maverick-17B-128E, zero-shot	20.0	10.1	13.4
	Llama 4 generated scripts, multiple calls with feedback	19.0	14.5	16.4

Table 20: Triple extraction performance comparison between non-LLM and LLM-based approaches. All numbers are in percentage (%).

Model number	Step.	CPU						GPU				DDR4 Memory support	TDP (W)	Socket	Released	Part number	Release price (USD)
		Cores	Clock (GHz)		L2 cache (MB)	Multi	V_{core}	Model	Compute units	Clock (GHz)							
			Base	Turbo						Base	Turbo						
X3216	Unknown	2	1.6	3.0	1	Unknown	Unknown	Unknown	4	0.8	—	1600	12–15	BGA (FP4)	June, 2017	OX3216AAY23KA	Unknown
X3418	Unknown	4	1.8	3.2	2	Unknown	Unknown	Unknown	6		—	2400				OX3418AAY43KA	Unknown
X3421	Unknown		2.1	3.4		Unknown	Unknown	Radeon 7	8		—					OX3421AAY43KA	Unknown

Subject	Predicate	Object
	...	
X3216	<– *CPU Clock (GHz)* –> Base	1.6
	...	
X3216	<– *GPU Clock (GHz)* –> Base	0.8
	...	

Figure 3: A triple annotation example with disambiguation information.

	Global	Triple-level			
	FM	EM	P_{EM}	R_{EM}	F-1_{EM}
multiple calls with feedback, 3 iterations (i.e., Algorithm 1)	74.5	56.1	73.6	56.9	64.2
multiple calls with feedback, 2 iterations (i.e., change 3 to 2 in #6)	74.2	55.1	72.6	55.9	63.2
multiple calls with feedback, 1 iteration (i.e., change 3 to 1 in #6)	74.8	51.3	67.1	52.3	58.8
multiple calls with no feedback (i.e., remove #6–#10)	70.2	42.1	62.1	42.6	50.5
single call (i.e., remove #3–#11)	59.0	30.9	58.8	32.0	41.5

Table 21: Ablation tests for Algorithm 1 on cleaned pages.

Algorithm 1 GenerateScript(A, B)

Require: Two samples A and B, each containing HTML and the ground truth triples

Initialize collection of candidate scripts: $scriptCands \leftarrow \{\}$
Generate script using both samples and add to candidate scripts: $scriptCands \leftarrow scriptCands \cup \{\text{GenerateScriptFromLLM}(samples = \{A, B\})\}$
for each sample $s \in \{A, B\}$ **do**
    Generate sample-specific base script: $script_0 \leftarrow \text{GenerateScriptFromLLM}(samples = \{s\})$
    Add base script to candidate scripts: $scriptCands \leftarrow scriptCands \cup \{script_0\}$
    for $i \leftarrow 1$ to 3 **do**
        Execute previous script on sample's HTML: $execResultPrev \leftarrow \text{ExecuteScript}(script = script_{i-1}, html = s.html)$
        Generate improved script using feedback: $script_i \leftarrow \text{GenerateScriptFromLLM}(samples = \{s\}, previousScript = script_{i-1}, execResult = execResultPrev)$
        Add improved script to candidate scripts: $scriptCands \leftarrow scriptCands \cup \{script_i\}$
    end for
end for
return $\text{SelectBestScript}(scriptCands, evaluationSamples = \{A, B\})$

A.7 Comparison with traditional (non-LLM) triple extraction approaches

There are very few established open-source OpenIE systems. We benchmarked ReVerb (Fader et al., 2011), which, to our knowledge, is the only non-LLM system that can be directly applied to our task. Additionally, we evaluated the performance of REBEL (Cabot and Navigli, 2021). Although REBEL is not strictly an OpenIE system, it is trained to cover more than 200 of the most frequent relations in Wikidata. As shown in Table 20, the performance of both systems falls significantly short of zero-shot LLM baselines and script-based extraction.

A.8 Details of triple extraction through automatically generated scripts

Algorithm 1 describes the steps of generating a script for in-domain triple extraction through multiple calls with feedback, where *GenerateScriptFromLLM* in #2 #4 and #8 employs Prompt 6, Prompt 5, and Prompt 7, respectively. For fast, low-cost iteration, we employed exact match defined in Appendix A.2 as the metric for script selection in *SelectBestScript*. We report ablation tests in Table 21. The out-of-domain triple extraction follows the same algorithm, except that it uses triples extracted by the 3-shot extraction model (based on Llama 3.1-70B-Instruct) instead of ground truth triples.