

000 DRIPPER: TOKEN-EFFICIENT MAIN HTML EXTRACTION 001 WITH A LIGHTWEIGHT LM 002

003 **Anonymous authors**

004 Paper under double-blind review

005 ABSTRACT

006
007
008
009
010
011 Accurately and efficiently extracting main content from general web pages is of
012 great significance for obtaining training data for large models. Using well-pre-
013 trained decoder-only generative language models offers excellent document com-
014 prehension capabilities, thereby effectively enhancing parsing quality. However,
015 it remains constrained by issues such as context window length, inference cost,
016 and format hallucination. We present Dripper, an efficient HTML main content
017 extraction framework powered by lightweight language models, which addresses
018 these challenges through four key innovations: (1) We design a specialized HTML
019 simplification algorithm that reduces input token count to 22% compared to raw
020 HTML while preserving critical structural information; (2) We reformulate main
021 content extraction as a semantic block sequence classification task, significantly
022 reducing inference cost; (3) We introduce a controlled decoding mechanism that
023 strictly constrains the output space through logits processors, effectively eliminat-
024 ing hallucination issues common in small-scale models; (4) We propose Main-
025 WebBench, an evaluation dataset containing over 7,800 web pages with metic-
026 ulously human-annotated main content extraction labels. Experimental results
027 demonstrate that using only a 0.6B parameter model, Dripper achieves state-of-
028 the-art performance across all evaluation benchmarks and outperforms all baseline
029 methods, attaining an ROUGE-N F1 score of 81.58%(83.13% with fall-back strat-
030 egy) on our proposed MainWebBench dataset.

031 1 INTRODUCTION

032
033 The World Wide Web forms the foundational data repository for modern AI, serving as the primary
034 source for training corpora like C4 (Raffel et al., 2020) and for building the knowledge graphs that
035 power large-scale applications (Wang et al., 2019). The sheer scale of this resource is immense, with
036 web archiving projects like Common Crawl (Common Crawl Foundation) preserving billions of new
037 pages each month. This massive volume presents a fundamental challenge for data utilization: the
038 raw, unstructured HTML must first be converted into high-quality, structured data. Accordingly, the
039 development of robust and accurate content extraction techniques has become a critical prerequisite
040 for a wide range of downstream information processing tasks (Vogels et al., 2018b).

041 The primary obstacle lies in the failure of traditional extraction methods to handle the web’s in-
042 herent complexity. While HTML standards provide semantic tags with clear intended uses—such
043 as `<article>` for main content or `<aside>` for sidebars—their adoption in practice is highly
044 inconsistent (Wang et al., 2022), rendering simple tag-based rules unreliable. Similarly, heuristic
045 methods based on statistical properties like text or link density often falter. Even pages built
046 from the same template can exhibit vast statistical variations simply due to differences in their core
047 content, undermining the stability of these metrics. Furthermore, vision-based approaches like diff-
048 bot (Diffbot, 2025) are often rendered ineffective in large-scale offline processing scenarios. Web
049 archives like Common Crawl typically store only raw HTML, lacking the corresponding CSS files
050 required to render a page as its developer originally intended. These fundamental challenges ex-
051 plain why established tag-based, heuristic, and vision-based methods struggle to achieve both high
052 accuracy and robust generalization. While the semantic understanding of well-trained decoder-only
053 language models offers a promising theoretical solution (Wang et al., 2025), their direct application
054 is thwarted by a distinct set of severe practical barriers. First, **excessive context length** makes pro-
055 cessing raw HTML infeasible at scale. Our analysis of 14,000 Common Crawl files shows 29.3% of

054 pages exceeded 32k tokens and 21.0% surpassed 128k tokens, lengths that far exceed the context
 055 windows of most SLMs. Second, the **structural complexity** of HTML presents a critical trade-off.
 056 While stripping all tags is an effective way to significantly reduce input length, this action simulta-
 057 neously destroys the vital structural information they contain. Without these cues, an algorithm
 058 cannot reliably distinguish main content from noise and perform accurate extraction. Finally, LLMs
 059 are prone to **output hallucination** (Ji et al., 2023), a tendency to generate content not present in
 060 the source document, which constitutes a critical failure for an extraction task that demands high
 061 fidelity.

062 To address these challenges, we introduce **Dripper**, a novel framework that reframes web content
 063 extraction as an efficient Sequential Block Classification task, specifically designed for Small Lan-
 064 guage Models (SLMs). Our three-stage pipeline begins with a pre-processing step that simplifies
 065 the raw HTML, making it tractable for a compact model. We then employ a 0.6B parameter SLM,
 066 **Dripper-0.6B**, to perform a localized binary classification on each semantic block of the simplified
 067 document. To ensure perfect output fidelity and eliminate hallucinations, we guide the SLM’s de-
 068 coding with a custom logits processor, forcing it to produce a structured sequence of labels. Finally,
 069 a post-processing step uses these high-confidence labels to precisely extract the corresponding con-
 070 tent blocks from the original HTML structure. The text from these selected blocks is then evaluated
 071 against the ground truth using ROUGE-N F1 as the primary metric. This approach circumvents the
 072 context length and hallucination issues inherent in holistic generative methods.

073 Our main contributions are summarized as follows:

- (1) We introduce **a novel HTML simplification algorithm** that strips redundant information while preserving critical structural markers, compressing the average document size by 22% and making processing feasible for SLMs.
- (2) The HTML document is represented as **a sequence of semantic blocks**, which transforms the task into a series of localized binary classifications. This approach dramatically reduces the problem’s complexity while retaining essential hierarchical and contextual relationships.
- (3) We design a **constrained decoding mechanism** using a custom logits processor. This converts the task from open-ended generation to producing a fixed, structured output, thereby systematically eliminating hallucinations and ensuring high-fidelity results.
- (4) To facilitate rigorous and comprehensive evaluation, we construct and will publicly release **MainWebBench**, a new large-scale benchmark with over 7,800 meticulously annotated samples, making it seven times larger than any existing public alternative. Our experiments demonstrate that Dripper, using only a 0.6B parameter model, achieves state-of-the-art performance, outperforming all baselines on MainWebBench with a leading F1 score of **81.58%**, which increases to **83.13%** when augmented with a fallback strategy. Our trained model weights¹, code² and the MainWebBench benchmark³ are publicly available.

092 2 RELATED WORK

094 Main text extraction aims to extract main content from raw HTML while filtering out boilerplate
 095 elements such as navigation and advertisements, a critical technique for building high-quality web
 096 corpora. The methods for accomplishing this task have evolved through several distinct paradigms,
 097 each addressing the limitations of its predecessor.

098 **Heuristic and rule-based Methods.** Early approaches predominantly relied on manually engi-
 099 neered heuristics to distinguish main content from boilerplate. These methods operate on the obser-
 100 vation that content-rich regions differ structurally from noisy elements, using features like text-to-tag
 101 ratios (CETR) (Weninger et al., 2010), visual cues from the rendered page (VIPS) (Cai et al., 2003),
 102 or a combination of heuristics such as link and stop-word density (Readability (Mozilla, 2015),
 103 jusText (Pomíkálek, 2011)). While computationally efficient, these methods are often brittle and
 104 require continuous maintenance to adapt to evolving web design patterns.

1¹<https://huggingface.co/anonymous-s2wrvq/Dripper>

2²<https://anonymous.4open.science/r/dripper-1825>

3³<https://huggingface.co/datasets/anonymous-s2wrvq/MainWebBench>

108 **Supervised Learning Methods.** To move beyond handcrafted rules, subsequent work approached
 109 body text extraction as a supervised machine learning problem. This paradigm shift began with
 110 classic methods like Boilerpipe (Kohlschütter et al., 2010), Dragnet (Peters & Lecocq, 2013), which
 111 treated the task as a classification problem using manually designed features. The advent of deep
 112 learning marked a further evolution from feature engineering to representation learning. (Vogels
 113 et al., 2018a; Leonhardt et al., 2020; Zhou et al., 2021). To better leverage the hierarchical structure
 114 of HTML, subsequent research introduced Graph Neural Networks (GNNs) (Zhou et al., 2021)
 115 and Transformer-based architectures like WebFormer (Endrédy & Novák, 2013), which improved
 116 extraction accuracy by capturing complex relationships between nodes. While achieving higher
 117 accuracy, these models often require substantial labeled data, and their complex architectures incur
 118 significant computational overhead.

119 **Hybrid Systems and Production Tools.** In parallel with academic advancements, a suite of pow-
 120 erful open-source tools has emerged, often blending multiple techniques for practical application.
 121 Trafilatura (Barbaresi, 2021) has become a strong baseline by integrating a sophisticated cascade of
 122 rules with established algorithms like jusText (Pomikálek, 2011) and Readability (Mozilla, 2015)
 123 as fallbacks. Other tools, such as magic-html (opendatalab, 2024), focus on simplifying complex
 124 HTML structures before extraction, often as part of larger document AI ecosystems. More recently,
 125 frameworks such as crawl4ai (UncleCode, 2024) have adopted an explicitly hybrid architecture,
 126 combining rule-based selectors, traditional machine learning, and Large Language Models (LLMs)
 127 to provide versatile solutions for AI data pipelines.

128 **Generative-Language-based Methods.** Recent months have seen rapid progress in decoder-only
 129 large language models. Base models pre-trained on massive, high-quality, and highly-diverse cor-
 130 pora have become the de-facto starting point for most NLP tasks. The most representative work in
 131 this line is ReaderLM-v2 (Wang et al., 2025), which frames main-content extraction as an HTML-to-
 132 Markdown translation problem. Starting from a 1.5 B-parameter Qwen2.5 checkpoint, the authors
 133 first extend the context window to 512 k tokens through continual pre-training, then fine-tune with
 134 supervised fine-tuning (SFT) and direct-preference optimization (DPO) to produce clean Markdown.
 135 This pipeline reuses the open-source model zoo and inference-acceleration stacks already available
 136 in the LLM community. Nevertheless, even the official best-practice implementation ⁴ still expects
 137 the full, un-pruned HTML page as input and generates the complete body text in one pass. This in-
 138 incurs heavy computational overhead and, during long-sequence generation, often produces unwanted
 139 artifacts such as repetitions or un-escaped HTML tags. Consequently, the potential of SLMs for
 extraction remains largely untapped.

140

141 3 METHODOLOGY

142

143 In this section, we detail the methodology of our Dripper framework. We begin in §3.1 with an
 144 overview of the system’s three-stage architecture. Next, in §3.2, we elaborate on the core pre-
 145 processing and post-processing modules that enable efficient extraction. We then formally define
 146 the task as a sequence labeling problem in §3.3. Finally, in §3.4, we introduce our constrained
 147 decoding mechanism, which uses a custom logits processor to eliminate hallucinations.

148

149

3.1 SYSTEM ARCHITECTURE OVERVIEW

150

151

152 The Dripper framework operates through a three-stage pipeline: pre-processing, SLM-based extrac-
 153 tion, and post-processing. As illustrated in Figure1, the system takes a raw HTML document as
 input and transforms it into a clean, structured Markdown output.

154

155

156

157

158

159

160

161

154 The process begins with the pre-processing module, which takes a raw HTML document and gen-
 155 erates two distinct representations. The first is a Simplified HTML, which is simplified and
 156 chunked. The second is a Mapping HTML, which is only chunked but otherwise unmodified. This
 157 parallel representation is crucial for ensuring the final extracted content remains a valid subtree of
 158 the original Document Object Model (DOM). The Simplified HTML is then passed to Dripper-
 159 0.6B, which identifies and labels the main content blocks. The decoding process is constrained
 160 by a custom logits processor to guarantee the structural integrity and correctness of the output
 161 format. Finally, in the post-processing stage, the Dripper-0.6B’s classification output is used to prune

⁴<https://huggingface.co/jinaai/ReaderLM-v2>

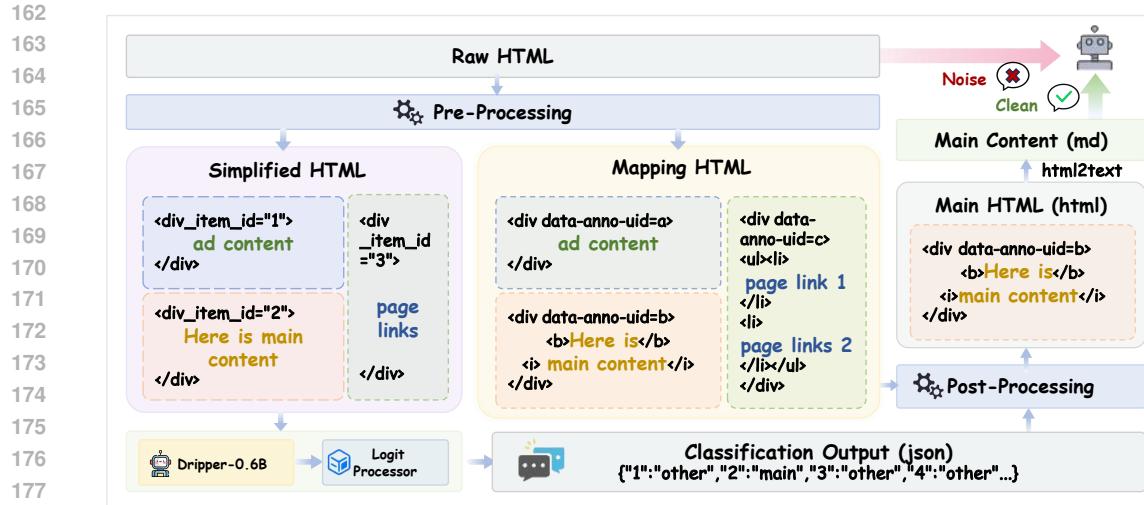


Figure 1: An overview of the Dripper framework, which operates as a three-stage pipeline. (1) Pre-processing: A raw HTML document is converted into two parallel representations: **Simplified HTML** for model input and **Mapping HTML** for final reconstruction. (2) Dripper-0.6B Extraction: Dripper-0.6B performs sequential block classification on the simplified input, guided by a custom logits processor to output a structured sequence. (3) Post-processing: The labels are used to select the corresponding blocks from **Mapping HTML** to construct the final, clean **Main Content**.

the **Mapping HTML**, yielding the final **Main HTML**. For downstream usability, **Main HTML** is converted into Markdown format using the `html2text`⁵ library.

3.2 PRE-PROCESSING AND POST-PROCESSING

Raw HTML is primarily designed for visual rendering, not for semantic interpretation by language models. Naively including all tags and attributes results in excessively long input sequences. Our pre-processing module is therefore guided by a multi-faceted strategy for simplification and chunking. The process begins with the **(1) preemptive removal of non-content tags**, such as `<style>`, `<script>`, `<header>`, and `<aside>`. Concurrently, we perform **(2) attribute simplification**, pruning all attributes except for class and id, which often carry the most valuable semantic cues for distinguishing content blocks. Following this, the document undergoes **(3) block-level chunking**, where it is segmented at elements that typically induce a line break in rendering. This strategy treats cohesive units like tables (`<table>`) and lists (``) as indivisible blocks to preserve their integrity. To handle the common misuse of tables for page layout, we apply heuristic rules to permit splitting within them when necessary. Finally, to manage excessively long individual blocks, such as a table with many cells, a list with numerous items, or an overly long paragraph, we employ **(4) partial content truncation**. For instance, we may retain only a subset of table cells or the initial 200 characters of a long paragraph, as we empirically find this partial data is sufficient for accurate classification while significantly reducing input length.

This pre-processing pipeline transforms Raw HTML into a sequence of simplified blocks ready for Dripper-0.6B. To ensure the final output is a valid DOM subtree, the **Mapping HTML** is generated in parallel by applying only the block-level chunking to the original, unmodified HTML. The post-processing module then uses the Dripper-0.6B’s output to select the corresponding content-bearing blocks from this **Mapping HTML** to construct the final result.

3.3 TASK FORMULATION

The system architecture detailed above effectively transforms the content extraction task into a well-defined **sequence labeling problem**. Formally, our pre-processing module converts an HTML doc-

⁵<https://pypi.org/project/html2text/>

216 ument into a sequence of n simplified blocks, $X = [x_1, x_2, \dots, x_n]$. Each block x_i has a corre-
 217 sponding ground-truth label $y_i \in \{0, 1\}$, where 1 indicates main content and 0 indicates boilerplate.
 218 The core task is to train a model f_θ that takes the sequence X as input and produces a predicted
 219 label sequence, $Y_{pred} = f_\theta(X)$, where $Y_{pred} = [y'_1, y'_2, \dots, y'_n]$. This predicted sequence is then
 220 used by the post-processing module to select the corresponding blocks from the Mapping HTML
 221 and construct the final Main HTML.

222 This sequence labeling formulation is highly efficient and reliable. By simplifying and chunking
 223 the input, the token load on the model is substantially reduced. Furthermore, framing the task as
 224 a classification of discrete blocks constrains the output to a simple sequence of binary labels. This
 225 design minimizes the required output length and, by avoiding free-form text generation, inherently
 226 eliminates the risk of hallucination, guaranteeing that the extracted content is a faithful subset of the
 227 original document.

228

229 3.4 CONSTRAINED DECODING VIA A CUSTOM LOGITS PROCESSOR

230

231 To eliminate hallucination and guarantee a valid output format, we implement a custom logits pro-
 232 cessor that functions as a deterministic finite state machine (FSM) during decoding. The FSM
 233 precisely controls the generation of the JSON-like output structure (e.g., `{"1": "main", ...}`) by de-
 234 terministically managing all syntactic tokens, such as braces, quotes, and numeric keys. At each
 235 decoding step, it masks the SLM’s logits, permitting the model to make a probabilistic choice only
 236 at the single critical juncture of classifying a block. At this point, the vocabulary is restricted to just
 237 ‘main’ and ‘other’, effectively converting the task into a series of high-confidence binary classifi-
 238 cations. This method guarantees syntactically perfect output, fundamentally removing the risk of
 239 format errors or extraneous content, and enables even a small 0.6B model to perform this structured
 240 prediction task with perfect fidelity.

241

242 4 DATASET AND BENCHMARK

243

244 In this section, we detail the construction of our large-scale training dataset (Section 4.1) and our
 245 new evaluation benchmark, MainWebBench (Section 4.2), along with its evaluation metrics.

246

247 4.1 TRAINING DATA CONSTRUCTION

248

249 To train our model effectively, we construct a large-scale, multi-faceted training dataset engineered
 250 to capture the diversity of the modern web. The dataset is curated through a three-stage sampling
 251 and filtering pipeline, ensuring variety in page layout, language, and document format.

252

Stage 1: Layout-Diverse Sampling. The initial stage focuses on capturing structural diversity. We
 253 begin by grouping pages by domain across 107 dumps of the Common Crawl dataset. For each do-
 254 main, we featurize the DOM tree structure of its pages (capped at 10,000 randomly sampled pages
 255 for larger domains) and computed their pairwise cosine similarity. We then apply the DBSCAN
 256 algorithm to these feature vectors to identify distinct layout clusters. From this process, we sam-
 257 ple one representative webpage from each of approximately 40 million unique clusters, yielding a
 258 candidate pool of 40 million structurally diverse pages.

259

Stage 2: Multilingual and Format-Aware Filtering. From this candidate pool, the second stage
 260 filtered for linguistic and format diversity. We first extract the main content of each page using
 261 Trafilatura and then employ Fasttext lid-176⁶ model for language identification. This
 262 step produced a balanced 10-million-page subset (4.75M English, 4.75M Chinese, 0.5M other lan-
 263 guages). To further enhance diversity, we categorize these pages using the format classifier proposed
 264 by (Wettig et al., 2025a). A final balanced sampling across these identified formats results in a set
 265 of approximately 1 million pages (485k English, 487k Chinese, 50k other) for the final annotation
 266 stage.

267

Stage 3: Final Annotation. In the final stage, we process these 1 million pages through our simpli-
 268 fication algorithm (detailed in Section 3.2). The resulting Simplified HTML is then provided to

269

⁶<https://fasttext.cc/docs/en/language-identification.html>

270 the Deepseek-chat API with a carefully crafted prompt (see Appendix Figure 6) to generate block-
 271 level labels. This automated pipeline yields approximately 1 million pages with high-quality, block-
 272 level annotations. After a final filtering step to remove samples containing no main content (i.e., all
 273 blocks were labeled as ‘other’), we obtain our final training dataset of 870,945 samples.
 274

275 4.2 MAINWEBBENCH: A NEW BENCHMARK FOR CONTENT EXTRACTION 276

277 To facilitate a more rigorous and fine-grained evaluation of web content extraction, we construct
 278 **MainWebBench**, a new benchmark comprising 7,887 meticulously annotated samples. Each sample
 279 contains four keys: ‘html’(the raw html document); ‘main.html’(the ground-truth as a valid
 280 html subtree identified by human annotators); ‘convert_main_content’(a Markdown representation,
 281 generated from the ground-truth); and ‘meta’(a rich set of annotations). MainWebBench is de-
 282 signed to serve as a gold-standard resource for evaluating extraction accuracy and enabling multi-
 283 dimensional performance analysis. An example data entry is shown in Appendix Figure 4.
 284

285 4.2.1 BENCHMARK CONSTRUCTION

286 MainWebBench is constructed using a hybrid sampling strategy to ensure broad representation: 90%
 287 of pages are randomly sampled from Common Crawl to cover the long-tail of the web, while 10%
 288 are drawn from a list of top-ranking websites (Chinaz Alexa⁷) to include popular, well-designed
 289 pages. To address the ambiguity in defining “main content,” we establish annotation rules based
 290 on two principles: **Contextual Integrity**, which includes content integral to the primary article
 291 (e.g., abstracts, references) and excludes peripheral elements (e.g., related-articles sidebars); and
 292 **Human-Generated Content**, which focuses on substantive material like article bodies and com-
 293 ments while filtering out auto-generated metadata (e.g., timestamps). Each page is meticulously
 294 annotated through a rigorous multi-stage process by using a custom-built tool(see Appendix Figure
 295 3). Furthermore, we enrich the benchmark with rich metadata annotations—including language,
 296 style, a quantitative difficulty level, and rich content tags—enabling fine-grained analysis. More
 297 details of benchmark construction can be found in Appendix A.5
 298

299 4.2.2 EVALUATION METRICS

300 To accommodate the two primary output formats of extraction tools—(1) raw Markdown text and
 301 (2) Main HTML document—we establish a standardized evaluation protocol. For the latter case,
 302 all Main HTML outputs are first converted to a canonical Markdown representation using the
 303 html2text library to ensure a fair and consistent comparison. The primary evaluation metric
 304 is the ROUGE-N F1 score, computed between the predicted Markdown and the ground-truth. We
 305 use the jieba tokenizer for all computations and set N=5. We specifically choose ROUGE-N instead
 306 of ROUGE-L, as the latter’s Longest Common Subsequence (LCS) algorithm has prohibitive com-
 307 putational complexity on the long documents in our benchmark, making ROUGE-N a more scalable
 308 and practical choice for evaluation.
 309

310 5 EXPERIMENTS

311 5.1 EXPERIMENTAL SETUP

312 **Supervised fine-tuning.** We employ the Qwen3-0.6B((Team, 2025)) model as our base model,
 313 which is the smallest model in the Qwen3 series, featuring a 32K context window and support for
 314 over 100 languages. Supervised fine-tuning is performed using the Llama-Factory((Zheng et al.,
 315 2024)) framework, training on the full set of 870K samples for a fixed total of 4 epochs. We use the
 316 last checkpoint as **Dripper-0.6B**.
 317

318 **Baseline Methods.** To comprehensively evaluate Dripper, we compare it against a diverse set of
 319 establish and state-of-the-art content extraction systems. Our comparison spans a wide spectrum
 320 of approaches, including classic heuristic and rule-based systems, supervised learning methods,
 321 production-grade hybrid tools, and recent large language model-based extractors. A detailed list and
 322 description of each baseline method is provided in Appendix, Table 4.
 323

⁷<https://malexa.chinaz.com/>

Evaluation Modes. To ensure a fair comparison across tools with diverse output capabilities, we established a clear evaluation protocol. We test every applicable output format for each tool and use a consistent suffix to denote the mode: `-HTML+MD` for tools that output an intermediate HTML which we convert to Markdown; `-MD` for tools that natively output Markdown; and `-TEXT` for tools that natively output plain text. Because Dripper cannot process inputs that exceed its context-length limit, we assign a score of 0 to such inputs. Following the practice of `Trafilatura`, which uses a fallback algorithm for parsing failures, we also test a version of our method, `Dripper_fallback`, which invokes `Trafilatura` for oversized inputs.

5.2 RESULT OF OVERHEAD REDUCTION

The computational cost of a decoder-only language model is primarily determined by the input and output sequence lengths, with its complexity approximated by Equation (1).

$$\text{Cost} \approx (L d (N^2 + MN + M^2) + L d^2 (N + M)) \text{ flops} \quad (1)$$

where L is the number of attention layers, d is the hidden-state dimension, N is the number of input tokens, and M is the number of output tokens. For Qwen3-0.6B we set $L = 28$ and $d = 1024$.

To quantify the efficiency gains of our approach, we compare its cost against a naive generative baseline. The baseline cost is estimated by using `Raw HTML` as input to generate the full Markdown content. For our method, we use `Simplified HTML` as input and the structured JSON classification as output. We measure the token lengths for both scenarios on the MainWebBench, and the results are detailed in Table 1.

Table 1: Computational overhead comparison on MainWebBench. The Pre-process column distinguishes the methods: `Without` denotes the naive baseline (generating full Markdown from `Raw HTML`), while `With` denotes the Dripper framework (predicting JSON labels from `Simplified HTML`). We report mean and median values for Input/Output token lengths and estimated inference cost (FLOPs), with the Ratio row demonstrating the efficiency gains of Dripper.

Pre-process	Input length (tokens)		Output length (tokens)		Cost estimate (flops)	
	mean	median	mean	median	mean	median
Without	44705.9	31987.0	2303.7	675.0	1.102×10^{14}	3.206×10^{13}
With	5734.5	3109.0	383.4	187.0	5.702×10^{12}	5.254×10^{11}
Ratio	12.83%	9.72%	16.64%	27.70%	5.18%	1.64%

The results reveal a substantial reduction in computational overhead. Our pre-processing pipeline dramatically shortens the input, reducing the mean token count to just 12.83% of `Raw HTML`, which is crucial for fitting within the model’s context window. Simultaneously, reframing the task to output a compact JSON classification reduces the mean output length to 16.64% of the full content. These two synergistic effects culminate in a remarkable reduction in computational load, lowering the mean inference cost to just 5.18% of the naive approach. This makes SLM-based content extraction not only feasible but also highly efficient and controllable.

5.3 RESULTS ON MAINWEBBENCH

We present the main performance comparison on our MainWebBench benchmark in Table 2. The results are broken down by various tracks, including difficulty levels and the presence of rich content.

The results clearly demonstrate that Dripper achieves state-of-the-art performance, significantly outperforming all baseline methods across every track. The standalone Dripper model achieves an overall score of 0.8182, surpassing the best baseline, `magic-html` (0.7091), by a large margin. Notably, Dripper shows exceptional strength on challenging content types where traditional methods falter, such as pages with tables, equations, and especially conversational layouts (0.8028 vs. 0.5766 for the best baseline). This highlights the robustness of our semantic, block-based classification approach.

378
379
380
381
382
383

Table 2: Performance comparison on MainWebBench (ROUGE-N F1). Methods are categorized by Mode: HTML+MD denotes tools outputting intermediate HTML converted to Markdown, while MD and TEXT denote native Markdown and Plain Text outputs, respectively. Results are stratified by Overall performance, Difficulty Level (simple, mid, hard), and specific Rich Content Types (subsets containing tables, code, equations, or conversational text).

name	mode	all	simple	mid	hard	table	code	equation	conversational
magic-html (opendatalab, 2024)	Html+MD	0.7091	0.7811	0.7095	0.6367	0.6681	0.8471	0.8470	0.4678
Readability (Mozilla, 2015)	Html+MD	0.6491	0.7370	0.6525	0.5570	0.5896	0.7774	0.7800	0.4608
Trafilatura (Barbaresi, 2021)	Html+MD	0.6358	0.7277	0.6391	0.5396	0.5505	0.6006	0.7327	0.5750
Resiliparse (Bevendorff et al., 2018)	TEXT	0.6233	0.7099	0.6283	0.5304	0.5473	0.6474	0.7829	0.5346
Trafilatura	MD	0.6237	0.7115	0.6279	0.5305	0.5400	0.5741	0.7168	0.5766
Trafilatura	TEXT	0.6049	0.6900	0.6088	0.5149	0.5271	0.5566	0.6955	0.5681
html2text (Swartz et al., 2025)	MD	0.5977	0.7499	0.5812	0.4678	0.5937	0.7729	0.7129	0.5494
BoilerPy3 (Riebold et al., 2023)	TEXT	0.5413	0.6347	0.5448	0.4434	0.4380	0.4833	0.6590	0.4695
GNE (Kingname et al., 2024)	Html+MD	0.5148	0.6477	0.4942	0.4098	0.4129	0.5495	0.6160	0.3296
news-please (Hamborg et al., 2017)	TEXT	0.5012	0.5399	0.5250	0.4307	0.4193	0.5118	0.6701	0.4073
jusText (Pomíkálek, 2011)	TEXT	0.4770	0.5132	0.5070	0.4010	0.3962	0.3779	0.6652	0.5222
BoilerPy3	Html+MD	0.4766	0.6443	0.4706	0.3174	0.3783	0.5532	0.6157	0.4103
Goose3 (Lababidi et al., 2025)	TEXT	0.4354	0.4514	0.4645	0.3808	0.3589	0.2900	0.6376	0.3064
ReaderLM-v2 (Wang et al., 2025)	MD	0.2264	0.3374	0.2078	0.1403	0.1801	0.2431	0.2927	0.1537
Dripper	Html+MD	0.8182	0.8837	0.8178	0.7536	0.7693	0.8368	0.8889	0.7671
Dripper_fallback	Html+MD	0.8399	0.9010	0.8392	0.7799	0.7964	0.8673	0.9067	0.8028

395

396

397 Additionally, due to limitations in preprocessing capacity and model generalization, Dripper occa-
 398 sionally fails to extract meaningful content from certain pages. We note that since Dripper follows
 399 a fundamentally different technical approach compared to rule-based systems like `Trafilatura`,
 400 its failures tend to be orthogonal to those of such systems. This allows for a straightforward fallback
 401 strategy: when Dripper returns no valid output, we use `Trafilatura` as a backup. With this
 402 mechanism, the combined system (Dripper_fallback) achieves an overall F1 score of 0.8399. This
 403 result indicates that our semantic approach not only establishes a new state-of-the-art on its own but
 404 can also be effectively combined with existing methods to improve robustness and coverage.

405

406

5.4 ABLATION STUDY

407

408 To analyze the data efficiency of our approach, we fine-tune the Qwen3-0.6B model on training
 409 subsets of increasing size: 2k, 5k, 10k, 100k, and 870k. We evaluate each resulting checkpoint on
 410 MainWebBench, from which we excluded samples whose simplified HTML exceeded our 32k token
 411 context window, as the standard Dripper model is designed to score 0 on such oversized inputs. This
 412 results in a performance gap of about 1.9% (0.818 for the full bench and 0.834 for the filtered bench).

413

414 To isolate the impact of our constrained
 415 decoding mechanism, we compare the per-
 416 formance of models trained with and with-
 417 out the custom logits processor. As shown
 418 in Figure 2, the logits processor provides
 419 a consistent performance improvement
 420 across nearly all data scales. The
 421 most significant gain (+2.3%) is observed
 422 at the 2k data scale, indicating that the
 423 FSM provides a strong structural prior
 424 that helps the model learn the task more
 425 efficiently in low-data regimes. As the
 426 training set grows, the model begins to
 427 learn the output format implicitly, and
 428 the performance gap narrows. Neverthe-
 429 less, the logits processor provides an ab-
 430 solute guarantee of a syntactically perfect,
 431 hallucination-free output. This ensures
 432 the output is always stable and machine-
 433 readable, preventing format errors that
 434 would otherwise disrupt downstream tasks
 435 and making the processor a critical component for production-level reliability.

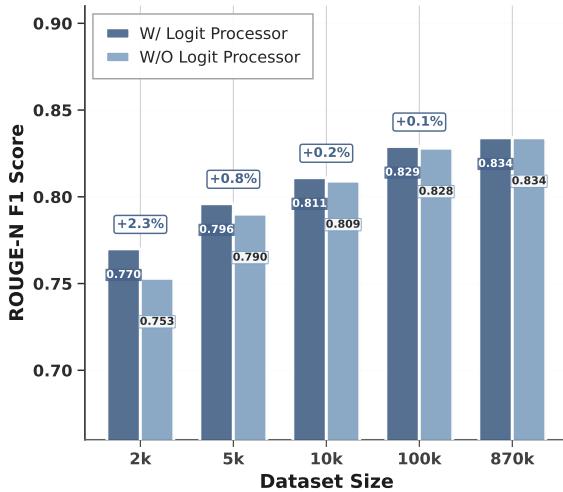


Figure 2: Impact of the logits processor on performance across various training data scales.

432 5.5 PERFORMANCE ON WCEB
433

434 To assess the generalization capabilities of Dripper, we evaluate it on the Web Content Extraction
435 Benchmark (WCEB, (Bevendorff et al., 2023)) , a comprehensive and unified benchmark. WCEB
436 addresses inconsistencies prevalent in many legacy datasets—such as plain-text-only ground truths,
437 file encoding errors, and corrupted content from script injections—by providing a filtered and stan-
438 dardized collection. Since the ground truths in this consolidated benchmark are in plain text, we
439 adapt our evaluation protocol by using the `html-text`⁸ library for the final conversion, a configu-
440 ration we denote as `Html+TEXT`. To enable a more granular analysis, we also apply our difficulty
441 stratification scheme to this dataset. A detailed description of the benchmark can be found in Ap-
442 pendix, Table 5.

443 The results on this suite of nine
444 established benchmarks, pre-
445 sented in Table 3, confirm Dri-
446 pper’s strong generalization ca-
447 pabilities. Our method again
448 establishes a new state-of-the-
449 art, with the standalone Dri-
450 pper model (0.8002) outper-
451 forming the strongest prior method,
452 `Trafilatura` (0.7833). Fur-
453 thermore, echoing the find-
454 ings on `MainWebBench`, the
455 `Dripper_fallback` strategy again
456 demonstrates the complemen-
457 tary nature of our SLM-based
458 approach and traditional heuris-
459 tics, boosting the score further
460 to 0.8154. This strong per-
461 formance across a diverse col-
462 lection of legacy datasets highlights
463 Dripper’s robustness, setting a
464 new state-of-the-art for general
465 web content extraction.

466 Table 3: Generalization performance on the WCEB dataset
(ROUGE-N F1). Given the benchmark’s plain-text ground truth,
467 Mode is defined as: `Html+TEXT` (converting extracted HTML
468 to text) or `TEXT` (native text output). Results are stratified by
469 Overall performance and Difficulty Level (simple, mid, hard) to
470 demonstrate model robustness across varying page complexities.

name	mode	all	simple	mid	hard
Trafilatura	TEXT	0.7833	0.8122	0.7785	0.7609
Trafilatura	<code>Html+TEXT</code>	0.7791	0.7896	0.7758	0.7731
Readability	<code>Html+TEXT</code>	0.7642	0.7744	0.7595	0.7601
magic-html	<code>Html+TEXT</code>	0.7506	0.7780	0.7573	0.7144
Goose3	TEXT	0.7272	0.7432	0.7312	0.7059
Resiliparse	TEXT	0.7225	0.7697	0.7052	0.6985
news-please	TEXT	0.7048	0.7051	0.7103	0.6970
justText	TEXT	0.6936	0.7445	0.6966	0.6389
BoilerPy3	TEXT	0.6221	0.6481	0.6468	0.5631
html2text	TEXT	0.6142	0.7273	0.6165	0.4982
BoilerPy3	<code>Html+TEXT</code>	0.6015	0.6532	0.6035	0.5474
GNE	<code>Html+TEXT</code>	0.5166	0.5138	0.5069	0.5323
ReaderLM-v2	TEXT	0.3077	0.3718	0.2928	0.2636
Dripper	<code>Html+TEXT</code>	0.8002	0.8293	0.8005	0.7707
Dripper_fallback	<code>Html+TEXT</code>	0.8154	0.8363	0.8143	0.7959

471 6 CONCLUSION
472

473 In this work, we introduce Dripper, a highly efficient and accurate framework for web content extrac-
474 tion. We demonstrate that our custom-trained 0.6B parameter Small Language Model, Dripper-0.6B,
475 achieves state-of-the-art performance by reframing the extraction problem. Our approach’s success
476 is rooted in three key technical contributions. First, our HTML Simplification Algorithm intelli-
477 gently strips redundant tags and attributes, drastically reducing the input token count while preserv-
478 ing essential structural cues. This simplified document is then processed through our novel Se-
479 quential Block Classification paradigm, which transforms the open-ended extraction task into a series
480 of simple, localized binary classifications. Finally, to guarantee absolute fidelity, our Deterministic
481 Logits Processor constrains the SLM’s output during the decoding phase, which completely elimi-
482 nates the risk of hallucination and ensures a syntactically perfect structured output. To rigorously
483 validate our method, we also construct and release `MainWebBench`, a new large-scale benchmark
484 of 7,887 samples, on which Dripper-0.6B proves its superiority over all baselines. Furthermore, by
485 integrating a heuristic-based fallback for inputs that exceed its context window, our Dripper_fallback
variant pushes performance even higher, demonstrating the robustness and complementary nature of
our method.

⁸<https://pypi.org/project/html-text/>

486 7 LIMITATION AND FUTURE WORK
487488 Despite careful web preprocessing development, 1.3% of Common Crawl pages still exceed
489 Qwen3’s content-window limit post-simplification and remain unprocessable. Additionally, extreme
490 DOM structures in some pages break chunking/simplification algorithms, hindering effective main
491 text extraction. Future fixes include enhancing preprocessing and extending the base model’s con-
492 text window via continued pre-training (to relax preprocessing’s token budget). Moreover, while
493 we use Qwen3’s smallest 0.6B model to cut overhead, scaling to 100B-scale pages poses cost is-
494 sues. A promising solution is tailoring data recipes for web parsing to pre-train small (0.01B–0.1B)
495 dedicated base models from scratch, lowering inference costs.
496497 8 REPRODUCIBILITY STATEMENT
498499 We are committed to ensuring the full reproducibility of our research. The architecture of our pro-
500 posed framework, **Dripper**, and its core components are detailed in the Methodology Section 3.
501 The construction of our large-scale training dataset is described in Section 4.1, while the creation
502 and structure of our new benchmark are detailed in the **MainWebBench** Section 4.2. Our complete
503 experimental setup, including all baselines, evaluation protocols, and metrics, is presented in the
504 Experiments Section 5. To facilitate direct verification and future work, we have made our resources
505 publicly available: the full source code⁹, the trained **Dripper** model weights¹⁰, and the complete
506 **MainWebBench** benchmark¹¹.
507

508 REFERENCES

509 Anthropic. System card: Claude sonnet 4.5, 2025. URL <https://www.anthropic.com/news/clause-sonnet-4-5>.
510
511 Adrien Barbaresi. Trafilatura: A web scraping library and command-line tool for text discovery
512 and extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational
513 Linguistics and the 11th International Joint Conference on Natural Language Processing: System
514 Demonstrations*, pp. 122–131, 2021.
515
516 Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. Elastic ChatNoir: Search En-
517 gine for the ClueWeb and the Common Crawl. In Leif Azzopardi, Allan Hanbury, Gabriella Pasi,
518 and Benjamin Piwowarski (eds.), *Advances in Information Retrieval. 40th European Conference
519 on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York,
520 March 2018. Springer.
521
522 Janek Bevendorff, Sanket Gupta, Johannes Kiesel, and Benno Stein. An Empirical Comparison of
523 Web Content Extraction Algorithms. In *46th International ACM SIGIR Conference on Research
524 and Development in Information Retrieval (SIGIR 2023)*. ACM, 2023. doi: 10.1145/3539618.
525 3591920. URL <https://dl.acm.org/doi/10.1145/3539618.3591920>.
526
527 Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Vips: a vision-based page segmentation
528 algorithm. 2003.
529
530 Common Crawl Foundation. Common crawl: Open-source web crawl data & infrastructure.
531 <https://commoncrawl.org/>.
532
533 DeepSeek-AI. Deepseek-v3.2-exp: Boosting long-context efficiency with deepseek sparse attention,
534 2025.
535
536 Diffbot. Extract API: Structured Data Extraction, 2025. URL <https://www.diffbot.com/products/extract/>.
537
538 István Endrédy and Attila Novák. More effective boilerplate removal-the goldminer algorithm.
539 *Polibits*, (48):79–83, 2013.
540
541 ⁹<https://anonymous.4open.science/r/dripper-1825>
542 ¹⁰<https://huggingface.co/anonymous-s2wrvq/Dripper>
543 ¹¹<https://huggingface.co/datasets/anonymous-s2wrvq/MainWebBench>

540 Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A generic
 541 news crawler and extractor. In *Proceedings of the 15th International Symposium of Information*
 542 *Science*, pp. 218–223, March 2017. doi: 10.5281/zenodo.4120316.

543

544 Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang,
 545 Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM*
 546 *computing surveys*, 55(12):1–38, 2023.

547 Kingname et al. Generalnewsextractor, 2024. URL <https://github.com/GeneralNewsExtractor/GeneralNewsExtractor>.

548

549 Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow
 550 text features. In *Proceedings of the third ACM international conference on Web search and data*
 551 *mining*, pp. 441–450, 2010.

552

553 Mahmoud Lababidi et al. goose3, 2025. URL <https://github.com/goose3/goose3>.

554

555 Jurek Leonhardt, Avishek Anand, and Megha Khosla. Boilerplate removal using a neural sequence
 556 labeling model. In *Companion Proceedings of the Web Conference 2020*, pp. 226–229, 2020.

557 Mozilla. Readability.js, 2015. URL <https://github.com/mozilla/readability>.

558

559 OpenAI. Gpt-5 system card. openai.com/index/gpt-5-system-card, 2025.

560 opendatalab. magic-html. <https://github.com/opendatalab/magic-html>, 2024.

561

562 Matthew E Peters and Dan Lecocq. Content extraction using diverse feature sets. In *Proceedings of*
 563 *the 22nd international conference on world wide web*, pp. 89–90, 2013.

564 Jan Pomíkálek. Removing boilerplate and duplicate content from web corpora. *Disertacní práce,*
 565 *Masarykova univerzita, Fakulta informatiky*, 2011.

566

567 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi
 568 Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-
 569 text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.

570

571 John Riebold et al. BoilerPy3, 2023. URL <https://github.com/jmriebold/BoilerPy3>.

572

573 Aaron Swartz, Charlie Tanksley, et al. html2text, 2025. URL <https://pypi.org/project/html2text/>.

574

575 Qwen Team. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.

576

577 UncleCode. Crawl4ai: Open-source llm friendly web crawler & scraper. <https://github.com/unclecode/crawl4ai>, 2024.

578

579 Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. Web2text: Deep structured boilerplate
 580 removal. In *European Conference on Information Retrieval*, pp. 167–179. Springer, 2018a.

581

582 Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. Web2text: Deep structured boilerplate
 583 removal. *CoRR*, abs/1801.02607, 2018b. URL <http://arxiv.org/abs/1801.02607>.

584

585 Feng Wang, Zesheng Shi, Bo Wang, Nan Wang, and Han Xiao. Readerlm-v2: Small language model
 586 for html to markdown and json. *arXiv preprint arXiv:2503.01151*, 2025.

587

588 Peilu Wang, Hao Jiang, Jingfang Xu, and Qi Zhang. Knowledge graph construction and applications
 589 for web search and beyond. *Data Intelligence*, 1(4):333–349, 2019.

590

591 Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. Webformer:
 592 The web-page transformer for structure information extraction. In *Proceedings of the ACM Web*
 593 *Conference 2022*, pp. 3124–3133, 2022.

594

595 Tim Weninger, William H Hsu, and Jiawei Han. Cetr: content extraction via tag ratios. In *Proceed-
 596 ings of the 19th international conference on World wide web*, pp. 971–980, 2010.

594 Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini.
595 Organize the web: Constructing domains enhances pre-training data curation, 2025a. URL
596 <https://arxiv.org/abs/2502.10341>.
597

598 Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini.
599 Organize the web: Constructing domains enhances pre-training data curation. *arXiv preprint*
600 *arXiv:2502.10341*, 2025b.

601 Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyuan Luo, Zhangchi Feng, and
602 Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Pro-
603 ceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume
604 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguis-
605 tics. URL [http://arxiv.org/abs/2403.13372](https://arxiv.org/abs/2403.13372).

606 Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, and Sandeep Tata. Simplified dom trees for
607 transferable attribute extraction from the web. *arXiv preprint arXiv:2101.02415*, 2021.

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648 **A APPENDIX**
649650
651
652 **A.1 BASELINE METHODS FOR WEB CONTENT EXTRACTION**
653654
655
656
657
658 **Table 4: An overview of the baseline methods for web content extraction.**

659 Method	660 Description
Heuristic and Rule-Based Methods	
662 Readability	663 Reader view algorithm for removing distracting elements
jusText	Two-pass processing with block size, link density, and stopword heuristics
Goose3	Article extractor with hand-crafted rules
html2text	Simple HTML to markdown converter
GNE	Text and symbol density-based extraction using mathematical formulas
666 Resiliparse	667 Fast and robust heuristic extractor with HTML parsing
Supervised Learning Methods	
BoilerPy3	Python port of Boilerpipe, decision tree-based text block classification
Hybrid Systems and Production Tools	
Trafilatura	Sophisticated rule cascade with jusText and Readability as fallbacks
news-please	Meta-extractor combining multiple extractors for news articles
magic-html	HTML structure simplification for extraction pipelines
Pre-trained Language Models	
ReaderLM-v2	670 SLM-based content extraction with semantic understanding

671
672 **A.2 STANDARD BENCHMARKS**
673674
675 **Table 5: Details of the datasets that comprise the Web Content Extraction Benchmark (WCEB).**

676 Dataset	677 Pages	678 Source & Characteristics
CleanEval	738	679 De-facto standard dataset from 2007 shared task combining development and evaluation sets of English web pages with basic structural markup ground truth
CleanPortalEval	71	680 Extension of CleanEval featuring multi-page samples from 4 major news domains
CETD	700	681 Created for density-based extractor evaluation across 6 domains
Dragnet	1,379	682 Combined sources from popular RSS feeds, 23 major news sites, 178 Technorati blogs, plus CETD and CleanEval conversions
L3S-GN1	621	683 Created by BoilerPipe authors with unique HTML annotation using span-wrapped CSS classes for 5-level content relevance
Google-Trends-2017	180	684 Dataset created for BoilerNet neural network training featuring binary CSS class annotations on DOM leaf nodes to distinguish content from boilerplate
Readability	115	685 Mozilla reader mode test suite with original and simplified HTML for evaluation
Scrapinghub	181	686 Created by Zyte for benchmarking proprietary extraction services

involved: (1) an initial pass by one annotator, (2) a review and correction pass by a second annotator, and (3) a final quality assurance check by a senior inspector, who made the final adjudication to resolve any discrepancies. Pages uninterpretable due to rendering issues were discarded.

Metadata Annotation. To enable detailed, fine-grained analysis, we annotate each page with a rich set of metadata. This includes **Language**, identified by GPT-5(OpenAI, 2025) and labeled as `en` (English) or `non_en` (other), and **Style**, classified by GPT-5 as `Conversational` for pages with user-generated content or `Normal` otherwise. We also develop a quantitative **Difficulty Level**, determined by an `overall_complexity_score` calculated for each page. To compute this score, we first measure four distinct metrics: *DOM structural complexity* (based on tree depth and width), *text distribution sparsity* (transitions between text/non-text nodes), *content-type diversity* (a count of rich content types), and *link density* (the ratio of hyperlinked text). These four values are individually normalized, and their weighted sum produces the final score. Based on the distribution of this `overall_complexity_score` across the benchmark, we then categorize pages into `simple`, `medium`, and `hard` using the 30th and 70th percentiles as dynamic thresholds. Finally, we add **Rich Content Tags** to identify the presence of tables (`<table>`), code blocks (`<code>`), and mathematical formulas (`<math>` or LaTeX patterns) using BeautifulSoup.

A.6 DETAILED BENCHMARK STATISTICS

In this section, we provide granular statistics regarding the composition of MainWebBench to demonstrate its diversity and coverage. MainWebBench consists of 7,887 samples. As detailed in Section 4.2.1, the composition follows a hybrid sampling strategy: 90% are randomly sampled from Common Crawl to capture the “long-tail” of the web, while 10% are sampled from top-ranking websites to ensure the inclusion of popular, high-quality pages.

Domain Diversity. The dataset covers 5,945 unique domains, confirming that the data is not dominated by a few sources but possesses a high degree of diversity. Table 6 lists the top 10 domains sorted by sample count. Furthermore, the benchmark spans 150 distinct Top-Level Domains (TLDs), indicating a broad spectrum of global regions and website categories. The distribution of the top 10 TLDs is presented in Table 7.

Page Category Distribution. We utilized GPT-5 to classify the semantic type of every page in the benchmark. As visually demonstrated in Figure 5, the dataset covers a diverse range of page layouts, ranging from standard news articles to forums and product pages.

Language Diversity. The dataset includes web pages in 46 different languages. We present the partial language statistics (top 10) in Table 8. The complete statistical files have been uploaded to Hugging Face¹³.

Table 6: Partial Domain Statistics (Top 10 Sorted by Sample Count). This table highlights the variety in page styles, difficulty levels, and rich content elements even within the most frequent domains.

Domain	Count	Percent	Lang	Style	Level	Table	Code	Eq.
aniruddhadeb.com	39	0.49%	en	Article	simple	1	9	36
politics.stackexchange.com	30	0.38%	en	Forum	mid	0	0	0
www.ask.com	29	0.37%	en	Article	simple	1	0	3
en.wikipedia.org	27	0.34%	en	Article	hard	20	1	0
www.china.org.cn	23	0.29%	en	Article	simple	21	0	0
money.cnn.com	22	0.28%	en	Article	hard	18	0	7
data.epo.org	21	0.27%	en	Article	simple	21	0	0
m.weibo.cn	19	0.24%	zh	Forum	simple	0	0	0
spanish.china.org.cn	15	0.19%	es	Article	simple	14	0	0
china.org.cn	14	0.18%	en	Article	mid	13	0	0

A.7 PROMPT FOR DATA SYNTHESIS

¹³<https://huggingface.co/anonymous-s2wrvq/Dripper>

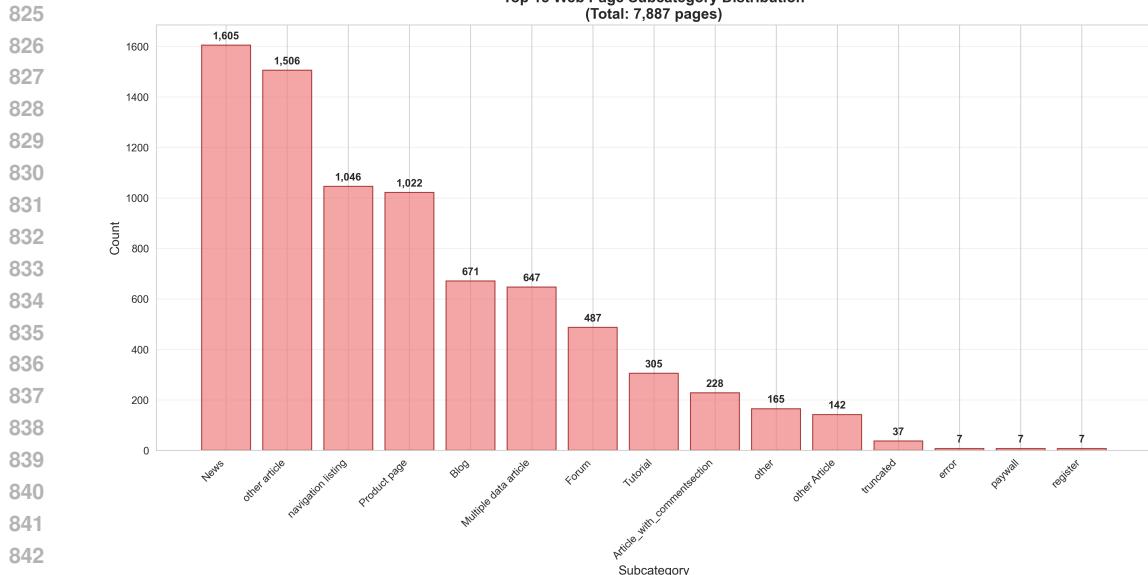
810

811 Table 7: Partial Top-Level Domain (TLD) Dis-
812 tribution (Top 10).

TLD	Count	Percent
com	4550	57.69%
org	816	10.35%
cn	459	5.82%
net	318	4.03%
uk	235	2.98%
edu	180	2.28%
de	101	1.28%
au	94	1.19%
ru	69	0.87%
gov	59	0.75%

823

824



839

840

841

842

843

844

845

Figure 5: Top 15 Web Page Subcategory Distribution. The types were classified semantically using GPT-5. The distribution shows a wide coverage from standard articles to forums and product pages.

846

847

848

849

A.8 ANALYSIS OF CLASSIFICATION METRICS

850

851

852

853

854

855

856

857

858

859

Table 9: Block-level classification metrics and ROUGE-N F1 across different training data sizes.

Data Size	Block-level Precision	Block-level Recall	Block-level F1	ROUGE-N F1
2k	0.877	0.781	0.756	0.770
5k	0.875	0.810	0.781	0.796
10k	0.888	0.823	0.800	0.811
100k	0.900	0.838	0.821	0.829
870k	0.898	0.843	0.826	0.834

864 While the data confirms our model’s strong classification capability, we deliberately chose ROUGE-
 865 N as our primary reporting metric for three key reasons.
 866

867 First, our pre-processing creates blocks with significant content length variance. A block can range
 868 from a single boilerplate word to a 2,000-word main article. Standard classification metrics treat all
 869 blocks equally, meaning a model could achieve a high F1 score by correctly classifying hundreds
 870 of tiny boilerplate blocks while missing the single, massive main content block. This would yield a
 871 high classification score but a completely failed extraction.

872 Secondly, ROUGE-N better aligns with the end-user’s objective, which is to obtain the complete
 873 main text. By measuring the overlap between the extracted text and the ground truth, ROUGE
 874 implicitly weights blocks by their information content, ensuring that the metric reflects the actual
 875 utility of the output.

876 Finally, prioritizing ROUGE-N ensures consistency with established benchmarks in the web extrac-
 877 tion literature, where ROUGE-L or ROUGE-N are the standard metrics for comparison.

879 A.9 PERFORMANCE COMPARISON OF LLM AND DRIPPER

880
 881 We compared performance of Dripper with GPT-5 (OpenAI, 2025), DeepSeek-V3.2-Exp
 882 (DeepSeek-AI, 2025) and Claude-Sonnet-4-5-20250929 (Anthropic, 2025) on the same input. Notably,
 883 our 0.6B Dripper model (0.8182) achieves 98.4% of the performance level of the state-of-
 884 the-art Claude-Sonnet-4.5 (0.8319). Although frontier LLMs exhibit a slight advantage in handling
 885 complex formatting tasks such as equations and conversational content, our Dripper_fallback strat-
 886 egy effectively bridges this gap, achieving an overall F1 score of 0.8399 that surpasses even the
 887 best-performing frontier models. Crucially, Dripper delivers this SOTA-level performance using a
 888 lightweight, locally deployable model, thereby avoiding the prohibitive latency and costs associated
 889 with querying massive frontier models for web-scale extraction.

890
 891 Table 10: Performance Comparison of LLM and Dripper

892 Model	893 All	894 Simple	895 Mid	896 Hard	897 Table	898 Code	899 Equation	900 Conversational
GPT-5	0.8302	0.8815	0.8301	0.7792	0.7957	0.8707	0.9161	0.7992
DeepSeek-V3	0.8252	0.8826	0.8244	0.7690	0.7804	0.8440	0.9113	0.8160
Claude-Sonnet-4.5	0.8319	0.8890	0.8329	0.7737	0.7919	0.8619	0.9273	0.8062
Dripper	0.8182	0.8837	0.8178	0.7536	0.7693	0.8368	0.8889	0.7671
Dripper_fallback	0.8399	0.9010	0.8392	0.7799	0.7964	0.8673	0.9067	0.8028

901 A.10 DETAILED PRE-PROCESSING ALGORITHM

902 In this section, we provide a comprehensive description of the HTML simplification algorithm,
 903 which serves as the cornerstone of the Dripper framework. The primary goal of this algorithm is to
 904 **drastically reduce the HTML token count while preserving the critical semantic and structural**
 905 **cues necessary for accurate content classification.** This is achieved through a multi-stage process
 906 applied to the raw HTML.

907 **1. DOM Cleaning and Pruning.** We first parse the HTML and proactively remove entire sub-
 908 trees known to be boilerplate. This includes tags such as `<script>`, `<style>`, `<header>`,
 909 `<footer>`, and `<nav>`. Furthermore, we heuristically remove elements whose `class` or `id` at-
 910 tributes contain keywords like ‘nav’, ‘footer’, or ‘header’, or which have CSS styles indicating they
 911 are hidden (e.g., `display: none`).

912 **2. Attribute Simplification.** To reduce noise and token overhead, we strip nearly all attributes
 913 from all elements. The only exceptions are the `class` and `id` attributes, which are often the most
 914 informative semantic markers in modern web design, and for `` tags, we also preserve the `src`
 915 (excluding large base64 data) and `alt` attributes.

916 **3. Semantic Block Segmentation.** The core of our method involves converting the cleaned DOM
 917 tree into a linear sequence of semantic blocks. We perform a recursive traversal of the DOM, seg-
 918 menting it at natural block-level boundaries. Our algorithm intelligently handles mixed content:

918 1. It identifies and preserves atomic block-level elements (e.g., a standalone paragraph or
 919 <div>).
 920 2. It aggregates consecutive inline elements (e.g., , links with text) and unwrapped
 921 text nodes into coherent blocks, wrapping them in a custom tag if necessary to maintain
 922 structure.
 923 3. It makes special provisions for complex structures like tables and lists, ensuring they are
 924 treated as single, indivisible units where appropriate.
 925

926 **4. Content Truncation within Blocks.** To handle excessively long blocks (e.g., a massive list or
 927 a very long paragraph), we apply a conservative truncation strategy. We recursively traverse the
 928 block’s content, limiting the total plain text to a predefined maximum length (e.g., 200 characters)
 929 while meticulously preserving the overall HTML tag structure. This ensures the model receives a
 930 representative sample of the content for classification without being overwhelmed by length.

931 **Parallel Generation Strategy.** A critical innovation in our pipeline is the parallel generation of
 932 Simplified HTML and Mapping HTML. Both representations undergo identical block segmen-
 933 tation, ensuring a one-to-one correspondence between blocks. However, Simplified HTML used
 934 for model input undergoes the full pruning and truncation process (steps 1-4). In contrast, Mapping
 935 HTML, used for final output reconstruction, undergoes only the initial cleaning (step 1) and segmen-
 936 tation (step 3), preserving the original, un-truncated

937 Finally, we inject a unique `_item_id` attribute to each block in both the Simplified and Mapping
 938 HTML. This allows the classification labels produced by Dripper-0.6B on the simplified sequence
 939 to be precisely mapped back to the rich, original content blocks for the final extraction.
 940

941 A.11 TRAINING CONFIGURATIONS

943 The specific hyperparameters and training configurations for supervised fine-tuning model are
 944 shown in table 11

946 Table 11: Supervised Fine-tuning (SFT) Configuration Details

947 Category	948 Parameter	949 Value
950 Model & Framework	951 Base Model	952 Qwen3-0.6B
	953 Fine-tuning Method	954 Full-parameter SFT
	955 Training Framework	956 LLaMA-Factory(Zheng et al., 2024)
957 Training Dynamics	958 Epochs	959 4
	960 Global Batch Size	961 128
	962 Max Sequence Length	963 32,000
964 Optimizer & Scheduler	965 Learning Rate Scheduler	966 Cosine
	967 Peak Learning Rate	968 1.00E-04
	969 Warmup Ratio	970 0.1
971 Hardware & Efficiency	972 Hardware	973 32× NVIDIA A100 GPU
	974 Precision	975 BF16

976 A.12 DOMAIN-SPECIFIC EVALUATION RESULTS

977 To comprehensively assess the generalization capabilities of our framework, we utilized the Topic
 978 and Format classifiers proposed by Wettig et al. (2025b) to categorize all 7,887 samples in the
 979 MainWebBench. Based on this classification, we calculated the ROUGE-N F1 scores for all methods
 980 across 24 distinct topics (e.g., Science & Tech, Finance, Health) and 24 distinct formats (e.g., News
 981 Article, Tutorial, Forum).

982 As shown in Table 12, Dripper_fallback consistently secures the 1st place ranking **across various**
 983 **topic and format categories**. On average, it outperforms the strongest existing baseline (magic-
 984 html) by approximately **19%**, demonstrating the exceptional robustness of this strategy. The stan-
 985 dalone Dripper model consistently ranks **second**.

972 These results consistently demonstrate our model’s strong capabilities across diverse domains. The
 973 complete breakdown of performance scores for every individual Topic and Format category is pre-
 974 sented in Tables 13 - 16.
 975
 976

977 Table 12: Average ROUGE-N F1 Scores and Rankings across Topics and Formats. Dripper and its
 978 fallback variant consistently outperform all baselines.
 979

(a) Topic Classification			(b) Format Classification		
Method	Avg Score	Rank	Method	Avg Score	Rank
Dripper_fallback	0.8583	1	Dripper_fallback	0.8320	1
Dripper	0.8330	2	Dripper	0.8102	2
magic-html	0.7209	3	magic-html	0.6974	3
readability	0.6829	4	readability	0.6342	4
trafilatura-html-md	0.6725	5	trafilatura-html-md	0.6259	5
resiliparse	0.6672	6	resiliparse	0.6122	7
trafilatura-md	0.6596	7	trafilatura-md	0.6144	6
trafilatura-text	0.6413	8	trafilatura-text	0.5956	8
html2text-md	0.6185	9	html2text-md	0.5827	9
boilerpy3-text	0.5811	10	boilerpy3-text	0.5377	10
newsplease	0.5595	11	newsplease	0.4944	12
gne	0.5499	12	gne	0.5049	11
justtext	0.5412	13	justtext	0.4766	13
boilerpy3-html-md	0.5229	14	boilerpy3-html-md	0.4676	14
goose3	0.4775	15	goose3	0.4321	15
readerlm	0.2492	16	readerlm	0.2205	16

1000 Table 13: Detailed ROUGE-N F1 Scores across Topics (Part 1/2).
 1001

Method	Home & Hobbies	Politics	Education	Software	Crime & Law	Science & Tech	Food & Drink	Social Life	Sports & Fitness	History	Finance	Literature
Dripper.fallback	0.7971	0.9119	0.8543	0.7890	0.8956	0.8519	0.8654	0.8547	0.8373	0.8334	0.8505	0.8107
Dripper	0.7993	0.8886	0.8301	0.7730	0.8558	0.8281	0.8452	0.8337	0.8210	0.7884	0.8340	0.7667
magic-html	0.6176	0.8139	0.7093	0.6505	0.8140	0.7433	0.6673	0.7102	0.7035	0.6983	0.7578	0.6751
readability	0.5147	0.7899	0.6439	0.6167	0.7555	0.6851	0.5963	0.6302	0.6464	0.6438	0.6922	0.6284
trafilatura-html	0.5430	0.7709	0.6344	0.6140	0.7475	0.6621	0.6360	0.6758	0.6129	0.6424	0.6657	0.6492
resiliparse	0.5570	0.7402	0.6238	0.5858	0.7341	0.6599	0.5997	0.6144	0.6058	0.6171	0.6420	0.5947
trafilatura-md	0.5283	0.7654	0.6216	0.5975	0.7346	0.6483	0.6474	0.6731	0.6102	0.6280	0.6572	0.6201
trafilatura-text	0.5084	0.7416	0.6010	0.5762	0.7166	0.6319	0.6140	0.6534	0.5951	0.6162	0.6357	0.6025
html2text-md	0.4830	0.6093	0.6032	0.6637	0.6065	0.6877	0.5093	0.5967	0.6031	0.6165	0.6310	0.5927
boilerpy3-text	0.5009	0.6802	0.5361	0.4822	0.6630	0.5417	0.5672	0.5559	0.5311	0.5284	0.5661	0.4851
newsplease	0.4599	0.5631	0.5055	0.4752	0.6435	0.4688	0.5134	0.4816	0.5240	0.4541	0.5474	0.4892
gne	0.4150	0.6554	0.5287	0.4422	0.6664	0.5294	0.4669	0.5335	0.5081	0.5099	0.5393	0.4806
justtext	0.5003	0.5195	0.4805	0.4236	0.6053	0.4322	0.5200	0.4454	0.4888	0.4387	0.5201	0.4683
boilerpy3-html	0.3806	0.5962	0.4893	0.4802	0.5605	0.5173	0.4603	0.4998	0.4573	0.4627	0.5189	0.4421
goose3	0.4256	0.4930	0.4312	0.3864	0.5748	0.3966	0.4815	0.4124	0.4661	0.4087	0.4785	0.3786
readerlm	0.1651	0.3002	0.2556	0.2045	0.3151	0.2485	0.2119	0.2357	0.2234	0.2113	0.2586	0.1899

1011
 1012 Table 14: Detailed ROUGE-N F1 Scores across Topics (Part 2/2).
 1013

Method	Health	Entertainment	Transportation	Hardware	Art & Design	Games	Fashion	Religion	Software	Dev	Travel	Industrial	Adult
Dripper.fallback	0.8806	0.8331	0.8320	0.8135	0.7774	0.7823	0.7526	0.8674	0.8420	0.8017	0.8386	0.7948	
Dripper	0.8555	0.8144	0.8124	0.7832	0.7756	0.7509	0.7315	0.8357	0.8156	0.7908	0.8219	0.7927	
magic-html	0.7776	0.6662	0.6716	0.6705	0.5899	0.6890	0.6061	0.7775	0.7299	0.6582	0.6930	0.6482	
readability	0.7219	0.6465	0.5678	0.5964	0.5314	0.6206	0.4388	0.7286	0.6631	0.6444	0.6218	0.5961	
trafilatura-html	0.7126	0.6314	0.5852	0.5818	0.5047	0.5971	0.5003	0.6762	0.5692	0.6133	0.5968	0.5980	
resiliparse	0.7031	0.6025	0.6071	0.5590	0.5157	0.5841	0.5112	0.6684	0.6051	0.6276	0.5663	0.5685	
trafilatura-md	0.6969	0.6212	0.5785	0.5628	0.5013	0.5853	0.4971	0.6665	0.5271	0.6023	0.5829	0.5930	
trafilatura-text	0.6782	0.6046	0.5571	0.5469	0.4837	0.5738	0.4755	0.6465	0.5123	0.5823	0.5643	0.5763	
html2text-md	0.6188	0.5826	0.5556	0.5254	0.4761	0.6154	0.3885	0.6758	0.7313	0.5124	0.5495	0.5514	
boilerpy3-text	0.6423	0.5238	0.5515	0.4783	0.5037	0.4785	0.4175	0.6242	0.3926	0.5793	0.5297	0.5465	
newsplease	0.5912	0.4901	0.5032	0.4410	0.4857	0.4316	0.3565	0.6161	0.4519	0.4845	0.4131	0.4749	
gne	0.6194	0.5116	0.4756	0.4428	0.4366	0.4467	0.3772	0.5425	0.4698	0.4671	0.5052	0.5473	
justtext	0.5685	0.4645	0.5030	0.4580	0.4644	0.4758	0.3988	0.5481	0.3159	0.5447	0.4004	0.4530	
boilerpy3-html	0.5584	0.4624	0.4471	0.3716	0.4052	0.4428	0.3018	0.5619	0.4264	0.4516	0.4295	0.4986	
goose3	0.5192	0.4299	0.4599	0.3936	0.3853	0.3804	0.3594	0.5662	0.3011	0.4487	0.3846	0.4083	
readerlm	0.2732	0.1949	0.2185	0.1930	0.1797	0.2047	0.1157	0.2638	0.1977	0.1964	0.1989	0.2366	

1026

1027

Table 15: Detailed ROUGE-N F1 Scores across Formats (Part 1/2).

Method	Comment	Section	Structured Data	About (Org.)	About (Pers.)	Tutorial	Product Page	Content Listing	Customer Support	User Review	Spam / Ads	News (Org.)	Knowledge Article
Dripper,fallback	0.7843	0.7310	0.8370	0.7547	0.9129	0.7971	0.6961	0.8392	0.8042	0.8178	0.9078	0.9047	
Dripper	0.7415	0.7020	0.8283	0.7229	0.9018	0.7865	0.6646	0.8348	0.7868	0.8039	0.9000	0.8986	
magic-html	0.4510	0.6045	0.7043	0.6254	0.8048	0.6425	0.5218	0.7598	0.5396	0.6960	0.8255	0.8142	
readability	0.4429	0.4765	0.6429	0.5260	0.7666	0.5365	0.4500	0.6371	0.5104	0.6965	0.7723	0.7870	
traflatura-html	0.5362	0.4261	0.6637	0.5967	0.7301	0.5448	0.4888	0.6849	0.5837	0.6487	0.7410	0.7355	
traflatura-md	0.5364	0.4201	0.6404	0.5783	0.7222	0.5245	0.4608	0.6583	0.5855	0.6358	0.7387	0.7216	
resiliparse	0.4794	0.4261	0.6177	0.5661	0.7260	0.5341	0.4932	0.6913	0.6074	0.5594	0.6908	0.7260	
traflatura-text	0.5311	0.3986	0.6296	0.5970	0.6895	0.5068	0.4486	0.6302	0.5377	0.6144	0.7277	0.7005	
html2tex-md	0.6010	0.6363	0.713	0.4970	0.6975	0.4573	0.5982	0.6274	0.4946	0.8010	0.5491	0.6227	
boilerpy3-text	0.4256	0.2503	0.6468	0.4808	0.6271	0.5128	0.2840	0.6163	0.5338	0.5295	0.6815	0.6333	
gne	0.2889	0.2810	0.6057	0.4517	0.5841	0.4470	0.2974	0.5013	0.4403	0.5613	0.6649	0.6256	
newsplease	0.2592	0.2387	0.5726	0.4586	0.6609	0.4437	0.3417	0.6157	0.5394	0.6320	0.5894	0.6461	
justtext	0.5093	0.1470	0.6460	0.4728	0.6483	0.4604	0.2739	0.6219	0.6065	0.1823	0.5098	0.5447	
boilerpy3-html	0.3933	0.2646	0.4876	0.4081	0.5856	0.3799	0.2572	0.5328	0.4197	0.4704	0.5436	0.5362	
goose3	0.2805	0.1509	0.5731	0.3824	0.6198	0.4131	0.2293	0.5034	0.4695	0.1697	0.5335	0.5632	
readerlm	0.1540	0.1168	0.2597	0.1493	0.3050	0.1556	0.1259	0.2683	0.1886	0.2269	0.2945	0.2647	

1036

1037

Table 16: Detailed ROUGE-N F1 Scores across Formats (Part 2/2).

Method	Academic	Writing	Personal	Blog	Creative	Writing	FAQ	Nonfiction	Writing	Truncated	News	Articles	Audio	Transcript	Legal	Notices	Documentation	Listicle	Q&A	Forum
Dripper,fallback	0.8943	0.9086	0.8913	0.9280	0.9756	0.7097	0.9535	0.9555	0.9462	0.8769	0.9423	0.8303								
Dripper	0.8444	0.8927	0.8609	0.9641	0.7164	0.9466	0.9093	0.9192	0.8174	0.9251	0.8207									
magic-html	0.8533	0.7698	0.8137	0.5514	0.9301	0.4891	0.8865	0.8881	0.9066	0.8082	0.8673	0.5470								
readability	0.7765	0.7291	0.7896	0.6627	0.9179	0.4756	0.8693	0.8519	0.8916	0.7552	0.8033	0.6228								
traflatura-html	0.7933	0.7141	0.7516	0.6959	0.8700	0.4748	0.8330	0.8526	0.7982	0.6028	0.7814	0.5921								
traflatura-md	0.7781	0.7193	0.7252	0.7175	0.8634	0.4678	0.8264	0.8322	0.7780	0.5510	0.7772	0.5816								
resiliparse	0.7767	0.7458	0.7129	0.7701	0.8894	0.4148	0.7675	0.8369	0.8034	0.6662	0.7737	0.7163								
traflatura-text	0.7628	0.7053	0.7050	0.6847	0.8484	0.4577	0.7973	0.8214	0.7589	0.5347	0.7521	0.5608								
html2tex-md	0.7734	0.6064	0.7053	0.6914	0.7577	0.3122	0.5856	0.8108	0.8701	0.8293	0.6909	0.4195								
boilerpy3-text	0.6720	0.6399	0.6485	0.6150	0.8390	0.4478	0.7804	0.7502	0.7463	0.4333	0.6640	0.4872								
gne	0.6584	0.5974	0.6908	0.4510	0.8065	0.4224	0.7529	0.7052	0.7880	0.4829	0.6907	0.4033								
newsplease	0.6260	0.6915	0.5124	0.6744	0.8476	0.3524	0.6324	0.7799	0.7123	0.4373	0.7317	0.6525								
justtext	0.6370	0.7155	0.4381	0.7084	0.8461	0.3432	0.5643	0.7810	0.6824	0.3567	0.7281	0.5653								
boilerpy3-html	0.6422	0.5575	0.6371	0.6178	0.7643	0.3265	0.6710	0.7428	0.8026	0.5231	0.6212	0.3654								
goose3	0.5321	0.6184	0.3610	0.6698	0.8174	0.3836	0.6142	0.7118	0.5075	0.3190	0.6972	0.3404								
readerlm	0.3292	0.2374	0.2802	0.2861	0.4097	0.1237	0.3553	0.3630	0.3996	0.2452	0.3138	0.1282								

1046

1047

A.13 USE OF LARGE LANGUAGE MODELS

1049

A large language model is used as a writing assistant during the preparation of this manuscript. The primary use of the LLM is for improving grammar, clarity, and phrasing of the text. The LLM does not contribute to the core research ideas, experimental design, data analysis, or the formulation of our conclusions. The authors have reviewed and edited all text and take full responsibility for the final content of this paper.

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

```

1080 f"""As a front-end engineering expert in HTML, your task is to analyze
1081     ↳ the given HTML structure and accurately classify elements with the
1082     ↳ {ITEM_ID_ATTR} attribute as either "main" (primary content) or
1083     ↳ "other" (supplementary content). Your goal is to precisely extract
1084     ↳ the primary content of the page, ensuring that only the most
1085     ↳ relevant information is labeled as "main" while excluding
1086     ↳ navigation, metadata, and other non-essential elements.
1087 Guidelines for Classification:
1088 Primary Content ("main")
1089 Elements that constitute the core content of the page should be
1090     ↳ classified as "main". These typically include:
1091         For Articles, News, and Blogs:
1092             The main text body of the article, blog post, or news content.
1093             Images embedded within the main content that contribute to the article.
1094             For Forums & Discussion Threads:
1095                 The original post in the thread.
1096             Replies and discussions that are part of the main conversation.
1097             For Q&A Websites:
1098                 The question itself posted by a user.
1099             Answers to the question and replies to answers that contribute to the
1100                 ↳ discussion.
1101             For Other Content-Based Pages:
1102                 Any rich text, paragraphs, or media that serve as the primary focus of
1103                 ↳ the page.
1104 Supplementary Content ("other")
1105 Elements that do not contribute to the primary content but serve as
1106     ↳ navigation, metadata, or supporting information should be
1107     ↳ classified as "other". These include:
1108         Navigation & UI Elements:
1109             Menus, sidebars, footers, breadcrumbs, and pagination links.
1110             "Skip to content" links and accessibility-related text.
1111             Metadata & User Information:
1112                 Article titles, author names, timestamps, and view counts.
1113                 Like counts, vote counts, and other engagement metrics.
1114                 Advertisements & Promotional Content:
1115                 Any section labeled as "Advertisement" or "Sponsored".
1116                 Social media sharing buttons, follow prompts, and external links.
1117                 Related & Suggested Content:
1118                 "Read More", "Next Article", "Trending Topics", and similar sections.
1119                 Lists of related articles, tags, and additional recommendations.
1120                 Task Instructions:
1121                     You will be provided with a simplified HTML structure containing
1122                         ↳ elements with an {ITEM_ID_ATTR} attribute. Your job is to analyze
1123                         ↳ each element's function and determine whether it should be
1124                         ↳ classified as "main" or "other".
1125                 Response Format:
1126                     Return a JSON object where each key is the {ITEM_ID_ATTR} value, and the
1127                         ↳ corresponding value is either "main" or "other", as in the
1128                         ↳ following example:
1129             {"1": "other", "2": "main", "3": "other"}
1130             Important Notes:
1131                 Do not include any explanations in the output, only return the JSON.
1132                 Ensure high accuracy by carefully distinguishing between primary content
1133                     ↳ and supplementary content.
1134                 Err on the side of caution, if an element seems uncertain, classify it
1135                     ↳ as "other" unless it clearly belongs to the main content.
1136
1137             Input HTML:
1138             {html_str}
1139
1140             Output format should be a JSON-formatted string representing a
1141                 ↳ dictionary where keys are item_id strings and values are either
1142                 ↳ 'main' or 'other'. Make sure to include ALL item_ids from the
1143                 ↳ input HTML
1144 """

```

Figure 6: Prompt template for Main HTML classification.