AutoClean: LLMs Can Prepare Their Training Corpus

Anonymous ACL submission

Abstract

Recent studies highlight the reliance of Large Language Models (LLMs) on high-quality, diverse data for optimal performance. The data sourced from the Internet often aggregated into datasets like the Common Crawl corpus, presents significant quality variability and necessitates extensive cleaning. Moreover, specific domain knowledge is usually presented in HTML, but there is a lack of effective methods to clean them into the training corpus automatically. Traditional cleaning methods involve either labor-intensive human teams that lack scalability or static heuristics that lead to suboptimal outcomes and are unable to be applied to specific target domains. In this paper, inspired by the recent progress in employing LLMs as versatile agents for diverse tasks, we take the initiative to explore the potential of these agents in automating data-cleaning methodologies. By configuring LLMs as an agent team that imitates the human data-cleaning team, we can automatically generate cleaning rules that traditionally require the involvement of data-cleaning experts. These rules are developed using a limited number of data samples and can then be applied broadly to substantial portions of raw data from the same domain. We demonstrate the efficiency and effectiveness of AutoClean on both pre-train scale corpora such as Common Crawl and specific target websites. Both automatic and human evaluations of the quality of the cleaned content highlight the feasibility of using LLMs to prepare their training corpus.

1 Introduction

The recent advent and swift advancement of Large Language Models (LLMs) (Brown et al., 2020) have marked a promising trajectory toward the realization of more generalized artificial intelligence. These models have now evolved to possess capabilities such as programming (Roziere et al., 2023) and following instructions (Wei et al., 2021). Consequently, these models are poised for deployment as agents (Wang et al., 2023a) capable of undertaking various human tasks, thereby liberating individuals from many tedious and time-consuming activities.

The training of LLMs currently faces a significant challenge due to the scarcity of high-quality data. According to the scaling law of LLMs (Kaplan et al., 2020), an increase in model size necessitates a corresponding increase in training data.

Two primary types of data sources are utilized in training LLMs. The first source is the vast volume of web content acquired through automated crawls, with the most notable corpus being the Common Crawl. Common Crawl is an extensive open-source repository of web pages. As of June 2023, it has accumulated approximately 11 petabytes of data[1] and continues to grow at a rate of about 200 to 300 terabytes per month[2]. However, this web-scale data is predominantly unrefined, with a significant proportion not immediately suitable for training LLMs due to quality concerns.

The second data source consists of specific domain repositories containing specialized knowledge on certain topics. Examples include a Chinese poetry website rich in classical Chinese poetry or a mathematical question-answering domain with high-quality mathematical reasoning corpora. These sources, however, lack automated methods for extracting cleaned text from noisy web content. Considering the dynamic nature of website content, it is even more crucial to accurately extract new information from time-sensitive websites.

The shortage of data from these two perspectives raises an urgent question: **how can we develop automatic methods to extract high-quality text from either vast web-scale data sources or specific domain websites?**

Various methods have been developed for auto-

[1] https://en.wikipedia.org/wiki/Common_Crawl

[2] https://commoncrawl.github.io/cc-crawl-statistics/

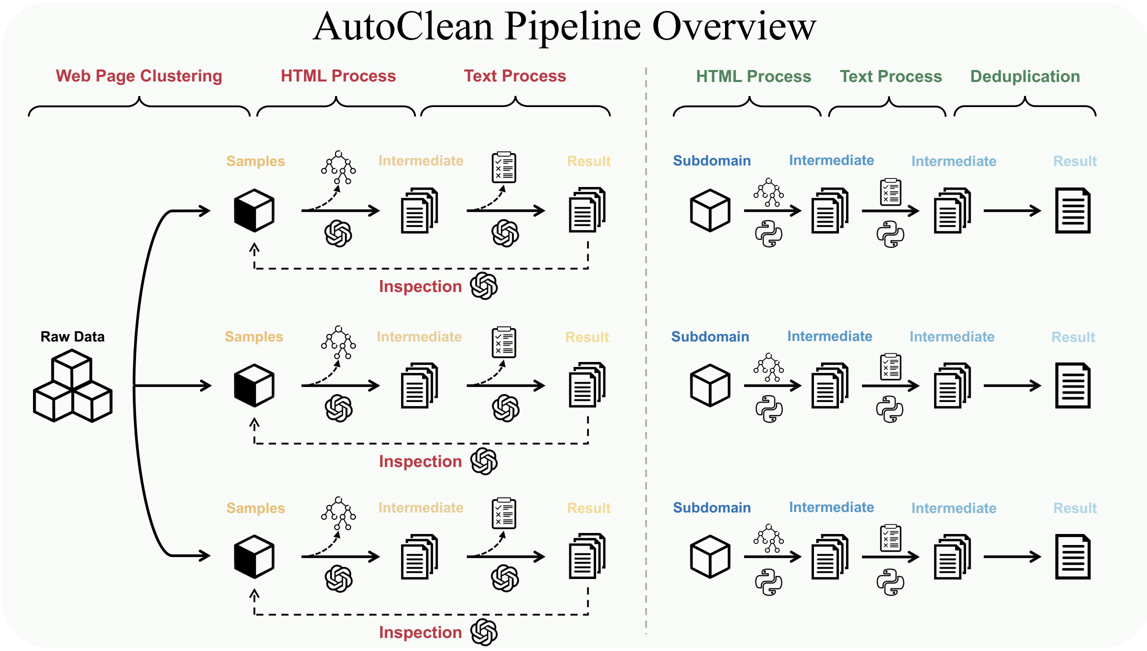

Figure 1: AutoClean consists of two parts: rule generation and data cleaning. The left part shows the cleaning rules generated by AutoClean based on the randomly collected samples, while the right part shows AutoClean cleaning the entire corpus according to the generated rules.

matically cleaning the Internet data. CCNet (Wenzek et al., 2020) employs a technique that involves deduplicating raw files, classifying file languages, and utilizing n-gram perplexity (PPL) to select high-quality data. Considering n-gram PPL is not always a reliable quality indicator, Pile (Gao et al., 2020) introduces a neural network to retain high-quality text, wherein raw data is first sorted by language types and then classified by a neural classifier. Similarly, RefinedWeb (Penedo et al., 2023) suggests a data-cleaning method with a sequence of deduplication, classification, and filtering pipelines. Despite the success of these methods, they implement fixed policies on highly variable raw corpora, leading to unpredictable and compromised outcomes in the data processing pipeline, highlighting the need for more intelligent and scalable approaches for data cleaning.

In this paper, we introduce AutoClean, which leverages LLMs themselves as the data cleaning agents, enabling intelligent and scalable data cleaning. AutoClean follows the recent advancements that conceptualize LLMs as autonomous agents (Qian et al., 2023a), capable of using tools to perform real-world tasks (Qin et al., 2023). At its core, AutoClean operates at the domain level, recognizing that web pages from the same domain often follow a similar structure. The LLM cleaning team generates a set of cleaning rules for a given domain by examining the sampled domain-specific web pages. The cleaning rules apply to all web pages belonging to this domain.

Specifically, the cleaning process begins with the HTML content of the website. Firstly, webpages are clustered into similarly structured subdomains. Then, an agent selects all potentially valuable HTML nodes. After collecting these nodes, a programmer agent applies a set of cleaning operations from a predefined set to further clean the web page. Finally, an observer agent evaluates the cleaned data to determine if it is suitable for direct use in training without further modification.

To validate the efficacy and efficiency of AutoClean, we conduct comprehensive experiments and analyses. We instantiate AutoClean with GPT-3.5 (OpenAI, 2021) as the agents and apply the rules generated by these agents to raw data from both Common Crawl and certain websites. Both automatic metric evaluation and human evaluation demonstrate superior data cleaning performance compared to previous heuristic methods.

To summarize, our contributions are as follows:

1. Design a pipeline that leverages Large Language Models (LLMs) for autonomous corpus cleaning.
2. Demonstrate the effectiveness of AutoClean in processing large-scale corpora and specific website cleaning.

3. Show through both automatic and human evaluations that our method achieves significantly cleaner text compared to heuristic approaches.

2 Related Work

Two lines of work are related to this paper: data cleaning methods, and agent automation.

2.1 Data Cleaning Methods

Before the advent of LLMs, datasets are predominantly manually curated for training task-specific models (Zhu et al., 2015; Zhang et al., 2015). However, the emergence of pre-trained language models necessitates larger datasets to facilitate the scaling of model sizes. Consequently, web-crawled data has become a prevalent solution (Kreutzer et al., 2022; Raffel et al., 2020; Dodge et al., 2021). Among such data sources, Common Crawl [3] stands out as the most extensive, forming the foundation for large-scale data corpora (Zellers et al., 2019; Trinh and Le, 2018; Penedo et al., 2023).

Web-crawled data often contains noisy and low-quality elements, such as programmatically generated content, promotional material, or unsafe content (Trinh and Le, 2018; Kreutzer et al., 2022). Many methods have been proposed to extract clean data from these web-crawled sources. The primary cleaning operations involve removing or down-weighting low-quality content. The distinction between cleaning methodologies largely depends on the criteria used for this process. An early approach, fastText (Grave et al., 2018), primarily employs simple deduplication and language filtering. CC-Net (Wenzek et al., 2020) utilizes PPL scores from statistical language models as the filter criterion. Additionally, heuristic rules, such as punctuation count have been explored for refining raw text corpora (Raffel et al., 2020). Pile (Gao et al., 2020) further utilizes a selector trained on OpenWebText2 to filter low-quality sections from Common Crawl. These refined datasets have been extensively utilized by various LLMs (Brown et al., 2020; Raffel et al., 2020).

However, these approaches are often based on heuristics that rely on substantial human labor and have limited flexibility, and scalability. In contrast, AutoClean is intelligent, scalable, and flexible, adept at handling the rapid emergence of web-crawled data.

[3] https://commoncrawl.org/

2.2 LLM Agent

LLM agents emerge as a promising avenue for LLMs to execute complex, real-world tasks. In this paper, we leverage two key aspects of agent automation. Firstly, we explore the concept of tool usage in LLMs. Innovations like AutoGPT (Significant Gravitas) and XAgent (XAgent, 2023) have enabled LLMs to access multiple APIs, performing multi-step operations to fulfill tasks. In AutoClean, we provide cleaning operations for LLMs.

The second feature related to AutoClean is multi-agent collaboration. This area has seen the development of numerous frameworks designed to efficiently and effectively simulate tasks involving multiple human-like agents (Hong et al., 2023; Chen et al., 2023; Wang et al., 2023b). These frameworks have been further refined and benchmarked in subsequent studies to enhance the role of LLMs in multi-agent collaboration (Liu et al., 2023; Qian et al., 2023b). Notably, ChatDev (Qian et al., 2023a) represents a landmark achievement in automating the software design pipeline by utilizing multiple agents to mimic the human software development process. Drawing inspiration from these advancements, AutoClean adopts a similar approach by simulating the human data-cleaning pipeline and achieves comparable results to human data-cleaning engineers.

3 Method

First, we introduce the top-level design of the AutoClean method. AutoClean begins by generating a set of cleaning rules. These rules are then applied to clean the entire dataset. It is worth noting that rule generation involves only a few randomly sampled web pages. However, the generated cleaning rules can be run on all web pages under the same domain without the need for agents, thereby achieving fast and low-cost cleaning of large-scale corpus. Next, we introduce the stages for generating the rules.

3.1 Web Page Clustering

AutoClean primarily leverages the similarity between different web pages under the same subdomain to clean the web pages. Hence the first step is to partition all web pages in a domain into subdomains. The desired outcome is that the web pages within each subdomain are highly similar. The similarity of a subdomain is defined by the similarity of randomly selected pairs of web pages within

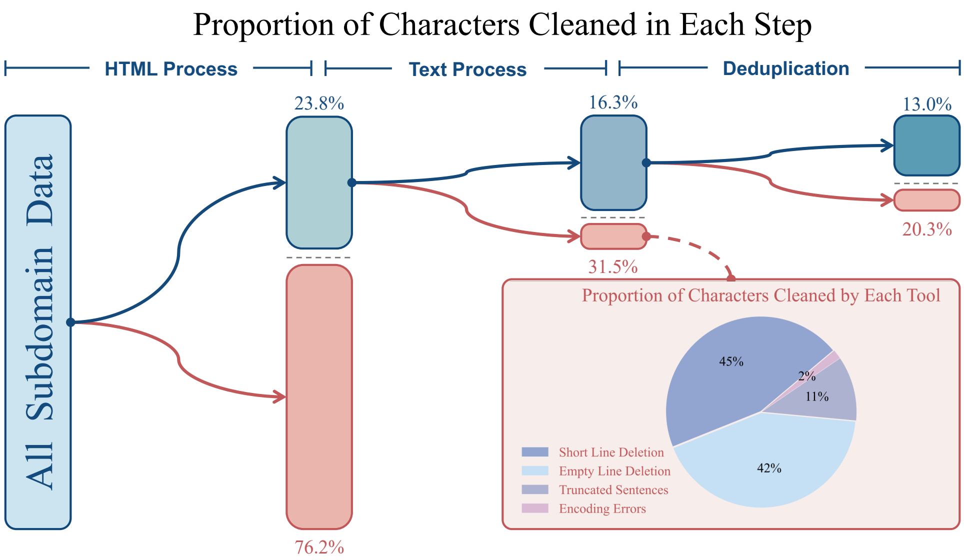

Figure 2: This figure shows the proportion of characters removed at each cleaning step. The blue/red numbers indicate the proportion of characters remaining/discarded. The pie chart illustrates the proportion of characters removed by each cleaning tool in the text process step.

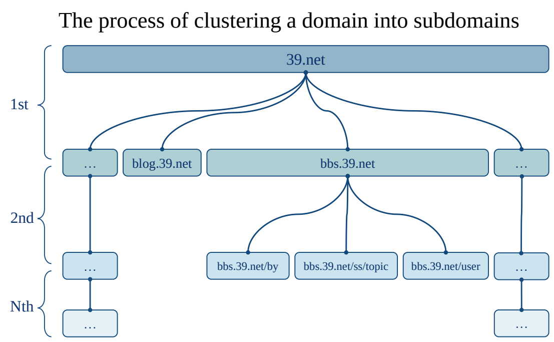

Figure 3: The process of web page clustering applying on 39.net. The nodes with insufficient similarity are divided into their child nodes. Leaf nodes represent the resulting subdomains.

the subdomain. For the similarity of webpages, they are deemed similar if the HTML nodes with a depth of less than 5 are identical. We use a recursive method to divide a large domain into highly similar subdomains, initially checking similarity, dividing by next-level domain names if needed, and merging smaller subsets during the process.

3.2 HTML Process

In this step, we utilize the Observer Agent to observe web pages, identifying nodes in the HTML structure tree that we wish to retain or delete. Based on a large amount of such data, we derive the Xpath paths where high-quality and low-quality texts are located for web pages in this subdomain.

3.2.1 HTML Observation

Node Quality Identification. We randomly sample some web pages for the Observer Agent to select nodes with high-quality and low-quality content. Specifically, we define leaf nodes as all nodes whose HTML tags are in a whitelist. The whitelist consists of all tags used to display text. And `<div>` nodes containing text directly are also leaf nodes. Then the Observer Agent will select the high-quality nodes from all leaf nodes, while the unselected leaf nodes are considered low-quality nodes.

Xpath Generation. We use two distinct strategies to identify high-quality and low-quality Xpath paths. Paths that lead from the root to all high-quality nodes are termed H-paths, while those leading to low-quality nodes are called L-paths. A path is considered valid if the number of H-paths using it as a prefix surpasses a certain threshold. For a path to be deemed high-quality, it must be valid and must not be a strict prefix of any other valid path. Conversely, a path is classified as low-quality if its occurrence among L-paths exceeds another threshold. For any web page within this subdomain, we start by removing all content associated with low-quality Xpath paths. From the remaining content, we then extract the portions under high-quality Xpath paths to complete the HTML processing for that web page

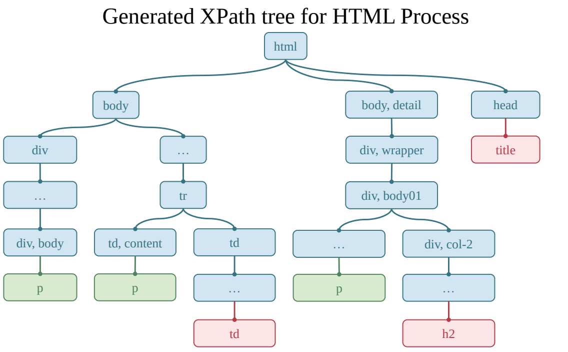

Figure 4: An example of the Xpath tree for a subdomain. The green/red nodes represent high-quality/low-quality Xpaths.

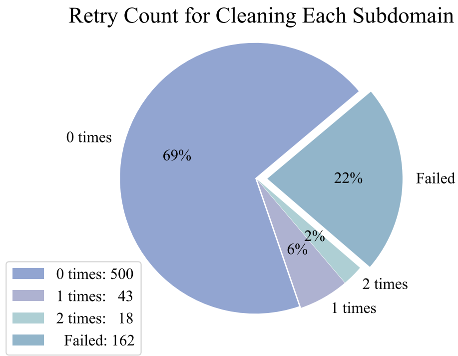

Figure 5: This pie chart describes the number of cleaning retries for all subdomains. The legend shows the subdomain counts in each category in the pie chart.

3.3 Text Process

In this phase, we adopt an observation-cleaning cycle to apply various cleaning tools 2. This phase is completed collaboratively by two agents. The Observer Agent first samples the dataset and generates an observation report detailing the current data quality issues. The Programmer Agent then reads the observation report and intelligently selects and applies some or all of the tools from the provided tool library to clean the data.

Observation Agent. This agent samples text from the last stage's result and generates an observation report based on a set of data quality criteria. If a text is too long, it will be split into multiple chunks, with each chunk summarized individually. The final summaries of all chunks from samples are then condensed into a single observation report, which is passed on to the Programmer Agent.

Programmer Agent. This agent reviews each cleaning tool by reading both the observation report and the tool usage instructions. Then this agent will determine whether each tool is applicable. Ultimately, all applicable tools will be added to the rules of this subdomain. Hence these tools will be applied to all web pages within this subdomain.

3.4 Quality Inspection

In this stage, we will inspect the results derived from the previous step. A portion of the web pages will be resampled, and the cleaning rules generated in the prior two steps will be applied. The Inspector Agent will evaluate the results obtained. Similar to the previous method, lengthy articles will be split into several chunks based on a fixed threshold. The Inspector Agent will then determine whether each chunk is closer to high-quality content or spam.

If the number of characters in a good chunk exceeds a certain proportion of the total number of characters, then this subdomain and its cleaning rules are valid. Otherwise, the agent will request a re-cleaning for this subdomain, starting over from the HTML process stage. If the number of retries becomes excessive, the domain will be deemed uncleanable and will be abandoned.

3.5 Deduplication

The web pages within a subdomain are highly similar, so we apply a line deduplication operation. Specifically, in each subdomain, we retain only the first occurrence of any completely identical text line, removing all subsequent duplicates.

Finally, we obtain a series of subdomains and their corresponding cleaning rules. By matching a web page's URL with the subdomain, we can apply the appropriate cleaning rules to extract high-quality textual data from the web page.

4 Experiments

In this section, we present the experiments. We conducted an experiment using AutoClean on Common Crawl and compared it with traditional pipelines to demonstrate the advantages of AutoClean.

Specifically, we run AutoClean on a 1TB[4] Common Crawl corpus containing 20 domains. All the steps described above are performed, resulting in a dataset approximately 14GB in size, containing 6,000,000 documents.

[4] Disk space, including the HTML scripts.

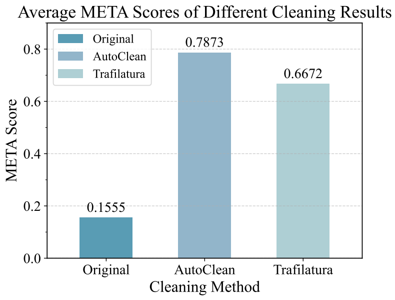

Figure 6: Average META Scores of Different Cleaning Results.

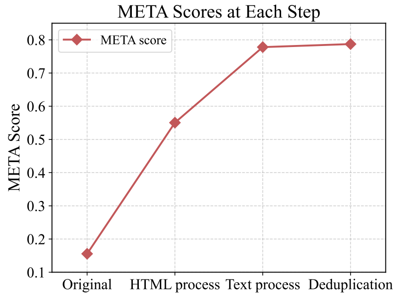

Figure 7: META Scores for intermediate results after different stages.

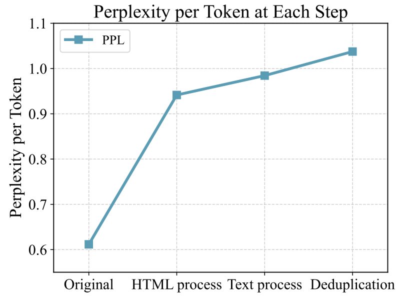

Figure 8: The average perplexity per token for intermediate results after different stages.

4.1 The Result of Web Page Clustering

723 subdomains are obtained through the web page clustering. Figure 3 shows the process of dividing one of the 20 domains into several subdomains. First, the domain 39.net is divided into several subsets according to the next level of domain name, including (blog.)39.net and (bbs.)39.net. The subset blog.39.net has already achieved sufficient similarity, so it stops splitting and becomes a subdomain. And bbs.39.net continues to be divided into three subsets. The leaf nodes represent the resulting subdomains derived from the domain.

4.2 The Result of HTML Process

In Figure 4, We demonstrated an Xpath tree to visualize high-quality/low-quality paths generated in the HTML process stage. These paths are rules used to extract high-quality content from web pages. The extraction will be operated on the HTML structure tree. Firstly, everything under the low-quality paths represented by red nodes will be removed. Then all contents under high-quality paths represented by green nodes will be extracted as the result.

4.3 Quality Inspection

Figure 5 shows the different outcomes of the 723 subdomains. Most subdomains met the quality requirements within three cleaning attempts, forming a rule set. 69% of the subdomains passed on the first attempt, while a small portion of subdomains passed within two retries. 22% of the subdomains did not pass after three attempts and were abandoned.

4.4 Dataset Quality Assessment by Human Experts

Trafilatura (Barbaresi, 2021) is a traditional method that extracts high-quality text directly from HTML. It obtains high-quality content by using a large number of empirical Xpath and regular expressions. To assess the quality of the dataset produced by our pipeline, 1000 web pages are randomly sampled from the 14GB dataset. We compare the cleaning result of AutoClean and Trafilatura (Barbaresi, 2021) on these web pages.

Web Extraction Issues	AutoClean	Trafilatura
Navigation Information	6.00%	30.33%
Irrelevant Information	3.00%	18.33%
Pagination Information	1.67%	2.67%
Top Navigation Bar	0.00%	4.67%
HTML Tags/Codes	0.67%	0.00%

Table 1: The percentages indicate the proportion of samples in the dataset that exhibit the issues described in each row.

Table 1 shows the results of comparing the cleaning effect on 300 web pages out of 1000 samples. The comparison method is human annotation. Annotation jobs are completed by 4 professional data annotators from large language model companies. The judgment criteria are based on a data cleaning standards manual which includes 26 rules, and approximately 2000 words, with over 60 illustrative examples. The relevant parts of the document are summarized in the appendix B.

Table 1 shows that our method significantly improves the removal of navigation bars and irrelevant information from web pages compared to the Trafilatura (Barbaresi, 2021) method.

4.5 Quantitative Dataset Quality Assessment

We also adopted the META method (Sharma et al., 2024) for a more comprehensive evaluation of our data cleaning effectiveness. The META (Sharma et al., 2024) method classifies high-quality corpus by scoring each corpus. It will firstly set a series of heuristic rules, then calculate the weight of these heuristic rules based on the text perplexity changes

Zhang Yan Biography - Ancient Poetry and Literature Network

Ancient Poetry and Literature Network
Recommendations
[…]
APP

Zhang Yan (1248-1320), courtesy name Shuxia, also known as Yutian and in his later years as Lexiao Weng. His ancestral home was Fengxiang, Shaanxi. […] , advocating the principles of "clarity" and "elegance." https://so.gushiwen.cn/authorv_c6454212c8da.aspx

~~Zhang Yan~~
Zhang Yan (1248-1320), courtesy name Shuxia, also known as Yutian and in his later years as Lexiao Weng. His ancestral home was Fengxiang, Shaanxi. […] , advocating the principles of "clarity" and "elegance." ► 361 poems ► 198 famous quotes
~~Improve~~
~~Style of Works~~
It is worth noting that Zhang Yan is the last significant author of Song lyrics. [..], his delicate and meticulous use of words and sentences often produced brilliant results. However, his excessive pursuit of poetic and picturesque details sometimes led to a lack of coherence in overall conception, resulting in a broad yet sometimes unfocused artistic realm
~~Expand to show all. ∨~~

~~Life~~
Zhang Yan (1248-1320?), courtesy name Shuxia, […] , is known as one of the "Four Great Masters of the Late Song Dynasty."
Zhang Yan, born into a noble family, enjoyed a leisurely life as a young nobleman for many years. […], leading to his family's decline. Fallen was
~~Expand to show all. ∨~~
► 361 poems ► 198 famous quotes

~~Login with WeChat~~
First-time users need to follow the public account | ~~Account login~~
Download the client to remove ads
© 2024 Ancient Poetry and Literature Network | Poetry | Famous Quotes | Authors | Ancient Books | Report Errors
Ancient Poetry and Literature Network Explanation

Figure 9: The selected page from www.gushiwen.cn for demonstrating in detail how AutoClean converts it into clean corpus. The omitted parts indicated by the ellipsis are similar to the adjacent ones. In high-quality/low-quality content, the ellipsis also represents high-quality/low-quality content.

caused by these rules. The weights are then used to score the corpus. We applied the META method (Detailed settings are in Appendix C) to evaluate 1000 samples. Text directly extracted by the HTML parser, as well as the cleaning results from AutoClean and Trafilatura are scored. The results in Figure 6 show that Trafilatura (Sharma et al., 2024) optimizes the quality, while AutoClean provides a better performance.

4.6 Analysis of Various Stages During Cleaning Process

META Score. We also performed a quantitative quality evaluation of the intermediate results in the AutoClean. It demonstrates the role of each stage in improving data quality. In Figure 7, it can be observed that after the HTML process, the data quality improves significantly, while the text process and deduplication also contributes to some improvement in data quality.

Perplexity. Perplexity, as a classic metric for measuring data quality in data cleaning, is also used to analyze the intermediate results in our experiments. We use the model from CCNet (Wenzek et al., 2020) to calculate perplexity per token. As shown in Figure 8, PPL increases after each cleaning stage. While the conclusion from CCNet (Wenzek et al., 2020) states that lower PPL indicates better corpus quality. Our analysis reveals that using PPL to evaluate data quality has significant flaws. This will be presented in an example in the Appendix 12.

4.7 The Cost of AutoClean

In this experiment, an average of approximately 1.71×10^6 GPT-3.5-Turbo tokens are used in each subdomain, and we handle a total of 723 generated subdomains from 20 domains. This means that generating a suitable rule for a domain costs around $40. According to our experiment, the 20 domains can obtain about 1.3% of high-quality data from same domain corpus in Common Crawl, especially considering the large volume of data in Common Crawl and the high quality of our method's cleaning results.

5 Domain Specific Data Acquisition

Despite being scalable for large corpus such as Common Crawl, our method has a special advantage over other methods when we target to acquire some specific type of data from a specific website.

Our method can be applied on any scale. While traditional methods like CCNet (Wenzek et al., 2020) require a larger scale to start. An important step in traditional methods, represented by CCNet (Wenzek et al., 2020), is to select the part of a large corpus set with the lowest PPL. When the corpus set is relatively small, the quality of web pages with relatively low PPL is not reliable. AutoClean can directly clean a domain on any scale while achieving good results. In this subsection, we will present examples.

We used a Python crawler to scrape the domain www.gushiwen.com and huggingface.co. We utilized our previous algorithm to determine whether two web pages are similar. Upon obtaining 100 web pages each two of them satisfy the similarity criteria. We use AutoClean to generate rules and clean the web pages.

We present a randomly selected sample in Figure 9 to visually demonstrate the cleaning procedure of AutoClean on this sample. First, in the HTML process stage, AutoClean utilizes the high-quality/low-quality paths to extract the parts highlighted in green from the entire web page while removing the parts highlighted in red. Then, the parts in bold and with strikethrough are removed using the short line deletion tool. The parts with blue underlines are sentences without punctuation marks and are removed using the truncated sentences tool. Both tools are selected by Programmer Agent in text process stage. Finally, the magenta lines are identical to their previous lines in the intermediate results and are thus deduplicated. It can be seen that most of the navigation bars and web components were removed.

Figure 13 shows the high-quality/low-quality paths generated on www.gushiwen.cn. This Xpath tree has the same meaning as Figure 4. Figure 10 represents the proportion of junk characters cleaned by each part in www.gushiwen.cn. It can be seen that the HTML process cleans most of the junk content, followed by short line deletion. Tools from the text process remove the remaining small portion of junk content. The result indicates the number of characters retained in the final cleaned data, accounting for 14.2% of the original corpus.

We repeated the quantitative data quality evaluation method from the previous section to demonstrate that AutoClean can also show sufficient effectiveness in this scale. In Figure 11, AutoClean demonstrated the highest quality scores on both the Chinese website www.gushiwen.cn and the English website huggingface.co. It can be seen that at the scale of 100 web pages, AutoClean is also highly competitive, achieving significantly higher scores compared to Trafilatura (Barbaresi, 2021).

We provided a Demo for this subsection, which can run on any domain to get clean corpus.

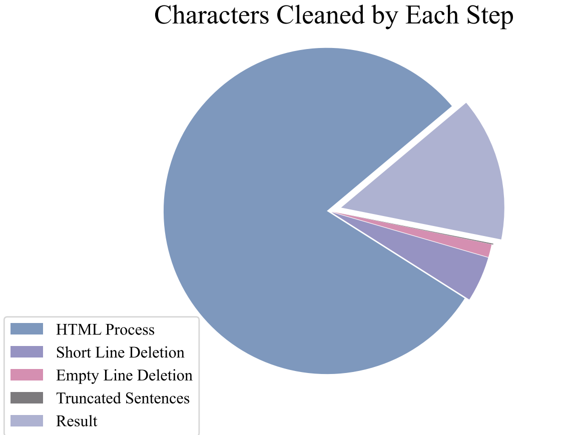

Figure 10: Characters cleaned by each part. The result represents characters retained in the end. Short line deletion, empty line deletion and truncated sentences are tools selected in text process.

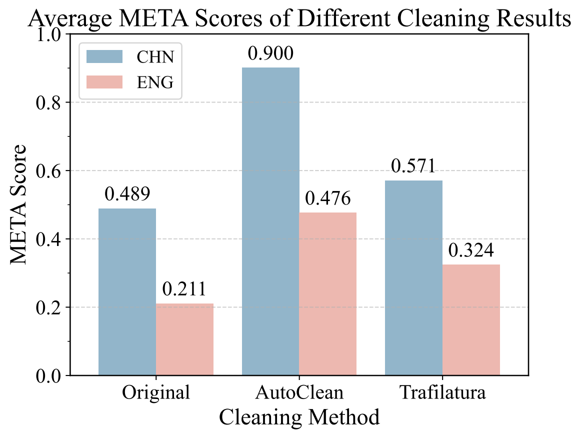

Figure 11: This figure shows the META scores on www.gushiwen.cn (marked as CHN) and huggingface.co (marked as ENG).

6 Conclusion

In this paper, we present AutoClean, a framework designed for automatic data cleaning by utilizing LLMs as agents. AutoClean generates a comprehensive set of cleaning rules using agents for each domain, thereby ensuring scalability, flexibility, and effectiveness. Future research directions include augmenting the AutoClean intelligence level to support more sophisticated data cleaning processes and distilling the capability of LLMs into smaller models to ensure effectiveness.

Limitations

There are several limitations of our work.

- The flexibility of the pipeline is somewhat constrained. Even though AutoClean generates rules intelligently, the pipeline adheres to a predetermined workflow, mirroring a typical data cleaning team's process. This could potentially limit the adaptability of the approach in diverse scenarios.
- The scale of the experiment presents another limitation. Owing to resource constraints, AutoClean has not been tested extensively with raw data corpus of Terabytes. The demonstration of its effectiveness is, therefore, restricted to a limited number of domains.
- We currently do not provide a direct comparison of model performances trained with corpus cleaned by AutoClean and other methods. However, we anticipate that a cleaner corpus will bring substantial performance improvement.

Ethical Considerations

In this paper, we present AutoClean, a novel data-cleaning workflow empowered by LLM agents. In the cleaning process of data, we currently do not include the step of screening for unsafe content, such as material exhibiting political bias. While there remains a possibility that the cleaned corpus may still contain such content, it is crucial to note that AutoClean's output comprises a set of rules for refining raw data, rather than content generated by the LLM itself. Consequently, AutoClean inherently avoids the introduction of additional unsafe content into the cleaned corpus.

The data source utilized in this study is open-source. During the comparative analysis between AutoClean and human data cleaning, the engineer tasked with refining four domains of the corpus was formally employed under our supporting affiliations, ensuring legal compliance.

We used GPT-4 as an tool for grammar correction in our paper writing.

References

Adrien Barbaresi. 2021. Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. volume 33, pages 1877–1901.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. *arXiv preprint arXiv:2104.08758*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Édouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomáš Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, et al. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

OpenAI. 2021. Introducing chatgpt.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023a. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2023b. Experiential co-learning of software-developing agents. *arXiv preprint arXiv:2312.17025*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Vasu Sharma, Karthik Padthe, Newsha Ardalani, Kushal Tirumala, Russell Howes, Hu Xu, Po-Yao Huang, Shang-Wen Li, Armen Aghajanyan, and Gargi Ghosh. 2024. Text quality-based pruning for efficient training of language models. *arXiv preprint arXiv:2405.01582*.

Significant Gravitas. AutoGPT.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023a. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023b. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. *arXiv preprint arXiv:2307.05300*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012.

XAgent. 2023. Xagent: An autonomous agent for complex task solving.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Tool Name	Usage
Encoding Errors	Identify and correct all characters with encoding errors.
Short Line Deletion	Delete all lines containing fewer than 20 characters.
Empty Line Deletion	Delete all lines that contain only spaces.
Adjacent Deduplicate	Delete adjacent lines that are completely identical.
Full-width to Half-width	Convert all full-width characters to half-width characters.
Truncated Sentences	Delete the last sentence if it does not end with a Chinese or English period to address the issue of text truncation.

Table 2: All tools provided in text process. The cleaning effects of each tool are described in the Usage column.

A Cleaning Tools Provided in Text Process

Table 2 presents all the tools provided to the Programmer Agent during the text processing stage. Each tool consists of a Python function and a textual instruction describing its usage.

B Data Cleaning Standards Manual

In this section, we explain the specific criteria for determining each web extraction issue listed in Table 1.

Navigation Information. There exists a list containing a series of hyperlinks. The texts in hyperlinks should include ellipses or be truncated. It's also suitable when the hyperlink texts include dates or comment numbers.

Irrelevant Information. There exist entire lines on the web page that do not relate to the main content. Irrelevant content inserted in lines related to the main content is not included in this category.

Pagination Information. "Previous page", "next page" and page numbers information used for navigating web pages appears in the cleaning result.

Top Navigation Bar. There is a navigation bar appearing at the top of a web page containing numerous categories. It usually includes many hyperlinked phrases for navigating between major categories.

HTML Tags/Codes. Information such as HTML tags and codes, like `[tag][/tag]` and `>` appears in the cleaning result.

C Heuristic Rules Used in META Method

We set up new heuristic rules to address different languages (META (Sharma et al., 2024) only provides heuristic rules for English). Table 3 and Table 4 show the heuristic rules we set for Chinese and English, respectively.

We use the corpus extracted directly by the HTML parser to calculate the all heuristic rules' weights. Then, we use these weights to score the results cleaned by AutoClean and Trafilatura (Barbaresi, 2021).

D Prompts

In this section, we list the prompts we used in each agent.

D.1 Observer Agent's Prompts

<TASK>:
Imagine you are a data cleansing engineer and now you are given a web page with some paragraphs and their HTML tags and asked to weed out low-quality content such as advertisements, buttons, page components, related recommendations, page sidebars, etc., and select semantically rich and coherent body paragraphs. Output their numbers, one number per line. If there are no semantic paragraphs, output "NONE".
{INPUT}

<TASK>:
Imagine you are a data cleansing engineer and now you are given a web page with some paragraphs and their HTML tags and asked to weed out low-quality content such as advertisements, buttons, page components, related recommendations, page sidebars, etc., and select semantically rich and coherent body paragraphs. Output their numbers, one number per line. If there are no semantic paragraphs, output "NONE".
{INPUT}

The two prompts above are used in section 3.2.1 to motivate the Observer Agent to select high-quality nodes.

<TASK>: Imagine you are a data engineer. Could you please check whether this text, which is the training corpus for a large model, contains low-quality content, incorrect punctuation, garbage short lines, and a host of other issues that degrade the quality of the corpus?
These issues specifically include but are not limited to:
1. the text contains redundant Markdown characters or has extra-long markdown reference paragraphs.
2. truncation problems in the text and semantic disjunctions at the end of the data.
3. extra line breaks, blank characters, wrong indentation, and other formatting problems.
4. incorrect use of punctuation, mixed use of full and half-width symbols, a large number of abnormal continuous symbols
5. irrelevant content in the paragraph, usually inserted advertisements or page components.
6. low-quality short lines, a large number of low-quality lines of short length in the article.
7. other problems.
Please output the problems you found in this text.
<TEXT>: {INPUT}

This prompt provides a standard for the Observer Agent to summarize the problem in one document.

<TASK>: Please summarize the following multiple reports on the issue of training corpus quality into a report, you only need to output the summary.
<REPORTS>: {INPUT}

This prompt is used to combine all summarized problems into a single report.

D.2 Programmer Agent's Prompts

<TASK>: Below you will be given a description of what a data cleansing tool does and a report of a problem with the existing data and asked to determine if the data should be cleansed using this tool, if yes please output YES, otherwise output NO.
<TOOL DESCRIPTION>: {INPUT}
<REPORT>: {INPUT}

The prompt above hints to the Programmer Agent to determine whether a tool is suitable for this subdomain.

Filter Name	**Description**
token_count_ge_3	Check if the token count is > 3.
word_count_3_256	Check if line word count is > 3 and < 256.
stop_word_match_2	Check if the line contains at least 2 stop words.
no_special_characters	Check if there is any '{' or '}'.
has_personal_pronoun	Check if there is any personal pronoun in the line.
terminal_punctuation	Check if the lines end with one of the Chinese punctuation marks.
digit_punctuation_ratio_0_25	Identify lines with a ratio of digits/punctuation to words in a line is > 0.25.

Table 3: Heuristic rules for the META method applying to Chinese corpus.

Filter Name	**Description**
has_noun	Check if the line has a noun.
has_determiner	Check if the line has a determiner.
token_count_ge_3	Check if the token count is > 3.
word_count_3_256	Check if line word count is > 3 and < 256.
stop_word_match_2	Check if the line contains at least 2 stop words.
no_special_characters	Check if there is any '{' or '}'.
has_object	Check if there is any object identified by the parser in this line.
terminal_punctuation	Check if the lines end with one of the English punctuation marks.
digit_punctuation_ratio_0_25	Identify lines with a ratio of digits/punctuation to words in a line is > 0.25.

Table 4: Heuristic rules for the META method applying to English corpus.

Zhang Yan (1248-1320), courtesy name Shuxia, also known as Yutian and in his later years as Lexiao Weng. His ancestral home was Fengxiang, Shaanxi. [...] , advocating the principles of "clarity" and "elegance.“

It is worth noting that Zhang Yan is the last significant author of Song lyrics. [...]. Due to his expertise in music theory, his delicate and meticulous use of words and sentences often produced brilliant results.

Zhang Yan (1248-1320?), courtesy name Shuxia, also known as Yutian and in his later years as Lexiao Weng. [...]. In literary history, he and another famous lyricist, Jiang Kui, are collectively known as "Jiang and Zhang." He, along with famous lyricists of the late Song Dynasty, Jiang Jie, Wang Yisun, and Zhou Mi, is known as one of the "Four Great Masters of the Late Song Dynasty.“

Zhang Yan, born into a noble family, enjoyed a leisurely life as a young nobleman for many years. In 1276, when Yuan soldiers captured Lin'an, Zhang Yan's grandfather, Zhang Ru, was killed by the Yuan forces, and their family wealth was confiscated, leading to his family's decline.

► 361 poems ► 198 famous quotes

Figure 12: The cleaning result of Figure 9.

D.3 Inspector Agent’s Prompts

> Imagine you’re a data cleansing engineer. You’re given a paragraph and asked to determine whether it’s more like a semantically rich and more coherent piece of text or more like the grossly incoherent garbage phrase content generated by page components, buttons, recommendations, sidebars, and other machinery. If it’s more like normal text, output “MAIN”; if it’s more like spammy phrases, output “SPAM”. Note that you only need to output "MAIN" or "SPAM".
> <PARAGRAPH>: {INPUT}

This prompt is used to enable the Inspector Agent to determine whether each paragraph is qualified. The percentage of qualified characters will be used to determine whether the subdomain is qualified.

E Case of High PPL with High-quality

In this section, we present an example that cleaned corpus has a higher PPL per token than the raw corpus with low quality. The raw corpus is all the content within the blue rectangle in Figure 9, while the cleaned corpus is shown in Figure 12.

The raw corpus has 2039 tokens with a total perplexity of 255.8, resulting in a PPL per token of 0.1255. While the cleaned corpus has only 1303 tokens and a total perplexity of 217.0, leading to a PPL per token of 0.1665.

It can be observed that the PPL per token of the cleaned corpus is higher than that of the raw corpus. However, by comparing Figure 9 and Figure 12, it is evident that cleaned corpus in Figure 12 contains less low-quality content such as navigation bars in Figure 9.

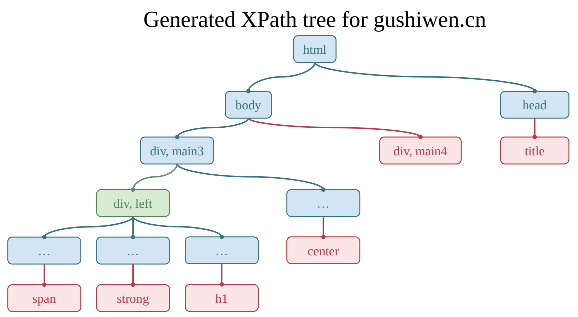

Figure 13: The high-quality/low-quality paths for www.gushiwen.cn with the same meaning as Figure 4.