ResponSet : A dataset highlighting responsive code in HTML/CSS

DESBIAUX Arthur[a,*,0], **VADUREL Benjamin**[b,**,0] **and SEN Abdurrahman**[c,***,1]

[a,b,c]University Claude Bernard Lyon 1

Abstract. This project consists of producing a dataset highlighting, in webpage code, what makes it responsive. The dataset is created using webpage code scraped from the 1000 most popular websites using the Tranco Ranking. Then, several LLM are used to extract the code that enables the responsiveness of the website.

1 Introduction

Nowadays, web development represents an important part of the IT activity. There are 1,11 billion [9] websites online with 71% of brands having a website. Front-end development usually takes a long time and developers face difficulties on making a website responsive. Code generation is currently a trending topic in this domain especially in front-end development. With this project, we aim to improve the capacity of LLMs to recognize the responsive parts in HTML and CSS code. This dataset could also be used to improve the capacity of LLMs to generate responsive code through fine-tuning.

2 Relevant links

A short demonstration video is available at https://youtu.be/irc_KAe42Jc and the code repository at https://github.com/Benjamin-Eldo/responset.

3 Dataset Architecture

The dataset is formatted in JSON and contains exactly 314 entries. Each entry is identified with the website domain name and is described by 3 fields : the HTML and CSS code for the webpage, and a list containing relevant code lines for the responsiveness of the webpage. The latter contains a list of code lines that can be either from the HTML or the CSS files.

```
{
  "website_id": <string>,
  "html_code": <string>,
  "css_code": <string>,
  "responsive_explanation": [<string>]
}
```

* Email: arthur.desbiaux@etu.univ-lyon1.fr
** Email: benjamin.vadurel@etu.univ-lyon1.fr
*** Email: abdurrahman.sen@etu.univ-lyon1.fr
[1] Equal contribution.

4 Dataset generation Pipeline

The dataset generation follows steps described in the pipeline illustrated in Figure 1. Each step must be completed before it passes to the next one. The pipeline contains 4 steps : Webpage scraping, Cleaning up the webpage code, Querying the LLMs on the webpage code and Response comparison.

Figure 1. Pipeline through which the dataset is generated

4.1 Webpage scraping

In order to ensure website code quality, the website selection for the dataset is based on the Tranco list generated on 13 December 2024 (Available at https://tranco-list.eu/list/KJ58W) [5]. The Tranco list contains a ranking of the most visited websites. For this dataset, only the top 1000 websites are used. Using the list, the dataset is created by scraping HTML and CSS code from the webpage using the Selenium library.

4.2 Cleaning webpage code

The resulting code from the previous step is large, with lots of links and content. Because link tags and website content do not serve website responsiveness, they are removed from the code. Additionally, some pages include non-latin characters, causing the language models to generate responses in non-English despite the prompt demanding otherwise. Thus, all non-latin characters have also been removed from the HTML and CSS files.

4.3 Querying LLMs

In order to build the dataset, the previously scraped and cleaned code is fed with a carefully crafted prompt to different language models.

Through the use of LLMs, the expectation is to be able to automatically annotate large datasets, thus allowing the processing of a larger quantity in this case of websites.

The expected output is of the following shape :

```
! `@media screen and (max-width: 768px)`
! `display: flex`
! `flex-direction: column`
! `justify-content: center`
! `align-items: center`
! `width: 100%`
! `margin: 0`
```

4.3.1 LLM Selection

An odd number of models are selected to elect the appropriate responses for the dataset through a majority vote. LLM choice is based on the HuggingFace [1] ranking and performance due to little computing resources. The following LLMs have been chosen :

1. Stable code:3b [8]
2. Qwen2.5-code:3b [4, 10]
3. Starcoder2:3b [7]
4. Gemma:7b [2]
5. Deepseek-coder:6.7b [3]

In order to generate the data, the prompt used includes the scraped HTML and CSS code. Selected models are then prompted for a response on each of the scraped website code. The responses are then saved to the dataset in the format shown previously.

4.3.2 Prompting

To teach the LLM what is expected as a response, Chain-Of-Thought prompting is used. An example with a simple responsive webpage and a list of code lines that make the page responsive is given to the LLM. Then, it is tasked as a *"web development expert"* to list down the important code element for responsiveness in the provided HTML/CSS code. The LLM is provided with a few instructions (presented in Figure 2) in order to format the responses such as separator and conciseness.

```
You're a web development expert, list down
the important code element for responsiveness
in each HTML/CSS code i give you.

Instructions :
1. Only give the line.
2. Each code piece must be separated by a '!'.
3. Do not give any explanation about the code.
4. The answers must be short and concise.
5. Your answers must be in english.
```

Figure 2. Prompt provided to LLM

4.4 Response comparison

The generated responses from any given language model are likely to miss information or include irrelevant (i.e. wrong w.r.t. responsiveness) ones. In order to reduce bias towards any single model, the responses from all 5 models will be compared to extract the common information, in a manner reminiscent of a tournament-style filtering.

4.4.1 Parsing

To compare model results, the text is processed in such a way that only pertinent information remains. Despite the prompt, there are different formats of responses for each LLMs. Different regex are used to only obtain the code in the responses. At the end, only a list of code snippets is obtained.

Here is an example of the response before the parsing for *gemma*:

```
! `@media (prefers-color-scheme: dark)` for
    dark mode responsiveness
! `@media (prefers-color-scheme: dark)` for
    dark mode typography adjustments
! `height: 76.391px` for responsive loading
    spinner
! `display: block` or `flex` for responsive
    layout elements
! `width: 100%` for responsive width
    adjustments
! `padding: 0` for responsive margin removal
! `position: relative` or `absolute` for
    responsive positioning
```

And after:

```
@media (prefers-color-scheme: dark)
@media (prefers-color-scheme: dark)
height: 76.391px
display: block` or `flex
width: 100%
padding: 0
position: relative` or `absolute
```

Due to observed lower performances from the Starcoder2 model (responses not aligning with given instructions at all), it was removed entirely from the following process.

4.4.2 Tournament

Once the LLM responses are parsed and formatted, similarity measurements are taken between the obtained responses. A tournament-like method is used to compare and filter LLM responses. There are two rounds in the tournament, and each round aims to confront two lists of responses. A round starts by creating pairs for each code line contained in the response lists and computing the rouge score [6] for each of the pairs. Using a threshold, code line pairs are filtered in order to keep only the most similar ones. As both code lines in the pairs are supposed to be similar, only the first one is kept for the next round. As described in the diagram (Figure 3), the first round confronts Stable-code and Gemma on the one hand, on the other hand Deepseek and Qwen-coder are confronted. The second round confronts the winners of the first round.

This method does not aim to provide the highest number of responses but ensure that the obtained responses are of sufficient quality.

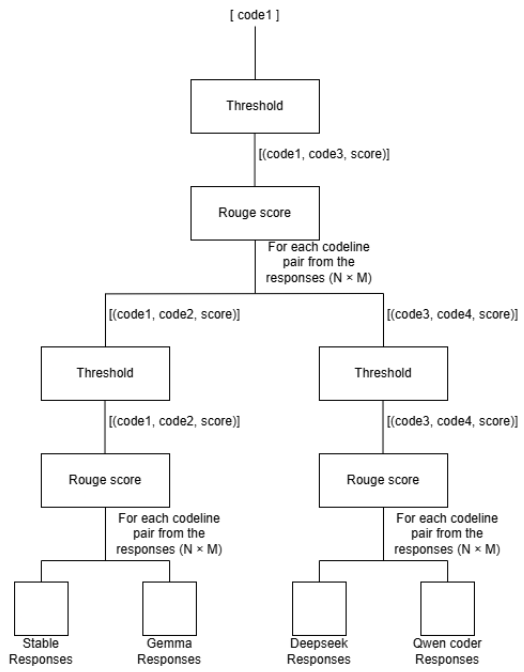

Figure 3. Diagram representing the tournament-style filtering

5 Results

5.1 Querying LLMs

Despite our best efforts to craft a prompt and clean the code fed to the LLMs, the responses obtained do not always align with the instructions. For example, here is a response from *deepseek-coder*:

```
The important code elements for
    responsiveness are as follows, separated
    by '!':
```html
    <meta name=\"viewport\" content=\"width=
        device-width, initial-scale=1.0\">!
    '@media (max-width: 600px) { nav { flex-
        direction: column; } nav a { padding:
         10px; }}'!
```
```

The response includes flavor text and places the exclamation points at the end of the line instead of the beginning. There are many more examples of responses from the same LLM varying in length, structure and content : sometimes the response deviates completely from the prompt and the LLM generates code for a jupyter notebook.

The reason for such a disparity with the expected output is surmised to be a result of the code snippets included inside the prompt being proportionally too long : the instructions are drowned out by code.

5.2 Response comparison

The following is an example entry from the final dataset, where the *responsive_explanation* is the result of our tournament-style filtering. The HTML and CSS codes have been removed for readability.

```
{
    "website_id": "arxiv_org",
    "html_code": "",
    "css_code": "",
    "responsive_explanation": [

        "@media screen and (max-width: 600px)
            { /* Styles for screens with a
            maximum width of 600 pixels */ }"
            ,
        ".flex-container { display: flex;
            flex-wrap: wrap; }"
    ]
}
```

The complete dataset is available for download here. From the 1000 web page entries we scraped, the dataset size has been gradually reduced to 314 entries. This is explained by the different filterings that occurred when undecodable characters were found in the scraped data during Querying and an empty result of the Response comparison both causing a skip of the entry.

There are a couple of avenues to improve the dataset size. The first is to scrap more websites. Another is to improve the prompting process to include less code for the LLM to better adhere to the instructions. This would improve the parsing that occurs down the pipeline.

References

[1] Big code models leaderboard - a hugging face space by bigcode, 2025. URL https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard.

[2] T. M. Gemma Team, C. Hardin, R. Dadashi, S. Bhupatiraju, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, and et al. Gemma. 2024. doi: 10.34740/KAGGLE/M/3301. URL https://www.kaggle.com/m/3301.

[3] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. K. Li, F. Luo, Y. Xiong, and W. Liang. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

[4] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Dang, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

[5] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, NDSS 2019, Feb. 2019. doi: 10.14722/ndss.2019.23386.

[6] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

[7] A. Lozhkov, R. Li, L. B. Allal, F. Cassano, J. Lamy-Poirier, N. Tazi, A. Tang, D. Pykhtar, J. Liu, Y. Wei, T. Liu, M. Tian, D. Kocetkov, A. Zucker, Y. Belkada, Z. Wang, Q. Liu, D. Abulkhanov, I. Paul, Z. Li, W.-D. Li, M. Risdal, J. Li, J. Zhu, T. Y. Zhuo, E. Zheltonozhskii, N. O. O. Dade, W. Yu, L. Krauß, N. Jain, Y. Su, X. He, M. Dey, E. Abati, Y. Chai, N. Muennighoff, X. Tang, M. Oblokulov, C. Akiki, M. Marone, C. Mou, M. Mishra, A. Gu, B. Hui, T. Dao, A. Zebaze, O. Dehaene, N. Patry, C. Xu, J. McAuley, H. Hu, T. Scholak, S. Paquet, J. Robinson, C. J. Anderson, N. Chapados, M. Patwary, N. Tajbakhsh, Y. Jernite, C. M. Ferrandis, L. Zhang, S. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries. Starcoder 2 and the stack v2: The next generation, 2024.

[8] N. Pinnaparaju, R. Adithyan, D. Phung, J. Tow, J. Baicoianu, and N. Cooper. Stable code 3b. URL https://huggingface.co/stabilityai/stable-code-3b.

[9] R. Vardhman. How many websites are there in 2024?, 2024. URL https://techjury.net/blog/how-many-websites-are-there/.

[10] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.