System Design for Text-to-SVG Generation in the "Drawing with LLMs" Competition

Carlos Andrés Brito Guerrero - 20241020147
Jose Fernando Ramirez Ortiz - 20241020080
Daniel Alonso Chavarro Chpatecua - 20241020066
Universidad Distrital Francisco José de Caldas
Bogota, Colombia

***Abstract*—This technical report presents a comprehensive system design for the "Drawing with LLMs" Kaggle competition, one that focuses on using Large Language Models to generate SVG images from text descriptions. The report details our analysis and possible design in which we would have a running program able to handle high-sensitivity components and incorporate continuous feedback loops for improvement. We observe the system's functional and non-functional requirements, propose technical implementations using open-source models, and provide strategies to address the identified sensitivity and chaos factors. Our design emphasizes modularity, performance optimization, and cost-effective implementation using Python-based frameworks and free-tier models like DeepSeek, Gemma, and Qwen. The system processes text descriptions (limited to 200 characters). Our approach aims to balance performance constraints at the same time of delivering high-quality SVG outputs that accurately represent textual descriptions that are inside of the competition's technical limitations.**

***Index Terms*—SVG generation, large language models, natural language processing, prompt engineering, open-source AI, generative AI, computer graphics**

I. Introduction

The union between natural language processing and computer graphics has given rise to exciting innovations, particularly in generative modeling. One emerging application is the generation of images from text prompts using Large Language Models (LLMs). The "Drawing with LLMs" Kaggle competition caves in and challenges participants to explore this frontier by designing systems that can convert short, 200-character textual descriptions into valid and meaningful SVG (Scalable Vector Graphics) images.

In this specific scenario, SVG synthesis requires precision and semantic understanding. Moreover, the challenge imposes computational and character constraints that demand efficient and adaptable solutions. In response, we propose an LLM able to solve the problem in such a compelling way, that in the near future, it could be considered human-like by some.

This paper documents our methodology, from system architecture and model selection to prompt engineering strategies and SVG evaluation metrics. Our main objective or goal is to contribute not only a working solution but also a framework for future developments in text-to-vector graphics as we said before.

II. Related Work

Prior research in text-to-graphics generation has primarily focused on quick uniform images, with models like Stable Diffusion, DALL·E, and MidJourney dominating the field. These systems excel at photorealism but struggle with structural precision. a critical requirement for SVGs. Some efforts, such as CLIPDraw and VectorFusion, attempt to overcome this problem by optimizing vector paths via differentiable rendering, but they often require optimization loops that are computationally expensive.

Our work positions itself by managing open-weight LLMs (avoiding proprietary APIs) and introducing postprocessing safeguards to ensure valid outputs. Unlike prior approaches that rely on multi-step raster-to-vector conversion, our method is end-to-end, prioritizing structural correctness and editability.

III. System design

Our SVG generation project is designed to balance flexibility and robustness, ensuring that outputs are both creative and technically possible and doable. The system will consist of several interconnected modules, each addressing a specific challenge in the text-to-SVG translation process. The first stage involves text preprocessing, where input prompts are normalized to remove ambiguities and expand shorthand descriptions (e.g., "red circle with a black border" instead of "red circle outlined") since these ones are the easiest to start with. This step helps reduce variability in model outputs and improves alignment between user intent and generated results.

The core of the system is the LLM-based generation stage, where a base model (such as DeepSeek or Gemma) synthesizes raw SVG code conditioned on the preprocessed prompt. We experiment with different test strategies, all that to tune the program in a way that's beneficial. Due to the complexity of SVG syntax, we enforce a strict 200-character limit on generated outputs, which encourages concise and efficient code while preventing overly verbose or convoluted structures.

After initial generation, the system performs syntax validation using XML parsers to detect and correct malformed markup, after all those validations, a scoring mechanism ranks outputs based on visual fidelity, prompt adherence, and structural simplicity, allowing the system to filter out low-quality results before presenting them to the user.

IV. Technical implementation

The implementation of our system is still highly hypothetical, however we have thought on using Python-based frameworks for scalability and ease of integration with existing AI tools. We would use the Hugging Face Transformers library for LLM inference, really useful for key aspects like cost-efficiency. For SVG manipulation, we would probably use libraries like svgwrite for programmatic generation and lxml for validation and parsing. These tools enable precise control over the generated vector graphics, ensuring compliance with SVG standards while allowing for dynamic modifications.

A critical aspect of our approach is model selection, where we evaluate multiple LLMs for their suitability in SVG generation. DeepSeek demonstrates strong performance in code synthesis tasks, plus, it is technically open source making it a natural candidate for structured outputs for a lower cost and better accesibility. Gemma on the other hand, being lightweight and efficient, offers a good balance between speed and quality, while Qwen provides more creative variations at the cost of increased computational overhead. We fine-tune prompts for each model to optimize output consistency, incorporating few-shot examples that illustrate desirable SVG structures.

V. Sensitivity and Chaos Factors

SVG generation is highly sensitive to variations in input phrasing, model architecture, and sampling parameters. Small changes in prompts—such as ”a smiling face” versus ”a cartoon smiling face”—can lead to drastically different outputs, ranging from simple circles to complex illustrations that the model will take more time to accomplish. This semantic fragility poses a challenge for reliable generation, as users may not always give precise descriptions and therefore, changing the desire result. One thing that we could mention, is the fact that inconsistency is at game too, since it is not unheared of to see where one same prompt can lead to very different results between each other.

To help out with these issues, we would implement several strategies. Model filtering generates multiple candidate SVGs for each prompt and selects the best based on scoring metrics that assess visual clarity, prompt alignment, and code validity. The program will probably take a hit in memory and performance, however, it would be better compared over being inconsistent or not lowering the amount of mistakes the program makes.

VI. Conclusion and Future Work

This paper shows us the analysis and design of an efficient framework for generating SVG graphics from short descriptions using open-source LLMs. Our architecture emphasizes accuracy and adaptability. Future work will help the program understand human context, among other things, all with the objective of furthering its efficiency and lowering down frecuency of mistakes.

Potential directions that the project could go on include the developing of fine-tuning models on SVG-specific corpora, developing better evaluation metrics for vector graphics, and extending the system to support dynamic SVGs with interactivity. As LLMs continue to advance, integrating geometric reasoning and user feedback mechanisms could further bridge the gap between natural language and structured visual design, unlocking new possibilities for creative automation.

Acknowledgments

We thank the organizers of the “Drawing with LLMs” Kaggle competition and the developers of the open-source LLMs and SVG libraries used in this study, as well as our teacher Carlos Sierra for the act of assinging us this homework