Understanding and creating data visualizations with LLMs

Sonja Gievska
Martina Tosevska
Dejan Ristovski

Faculty of computer science
and engineering - FINKI, UKIM

Abstract— **Chart generation and understanding is a problem that may seem quite easy to implement using LLMs. Although large models with high reasoning capabilities can easily address this problem, smaller models should not be overlooked and deserve evaluation for their chart generation capabilities. Can these models understand user's needs of chart visualization from just plain visualization description, and can they also then generate a suitable chart matching their needs? This paper strives to answer these questions by testing small LLMs that are open source, such as the LLaMA family of models and the Deepseek model. With specific prompt design and using state-of-the-art prompt techniques, the models are tested in different chart understanding and generation scenarios. The results reveal varying performance across the models, highlighting weaknesses in chart generation for some, while also demonstrating effective prompt techniques that improve outcomes in others.**

I. Introduction

With the growth of generative artificial intelligence and LLMs, more and more everyday tasks and problems can be automated. One of the most frequent usages of LLMs is their ability to generate code and give reasoning for their actions. This also shows their incredible ability to understand any request and use their general knowledge to help in almost any scenario. While most models used for research have 70B+ parameters, the small models have been quite neglected. As LLMs are starting to be integrated into almost any system, the research for the capabilities of small models is starting to raise.

Chart creation is a common requirement for data exploration and reporting, so automating this process could significantly reduce the technical barrier for non-expert users. In this paper we focus on the chart understanding and creation for LLMs with small parameter size. We test the models in scenarios where they need to generate a chart based only on the user's input, even without specifying the chart type that needs to be generated. This way we plan to simulate a real scenario where a non-expert may interact with an AI agent for dataset visualization.

In contrast previous works on this subject (Pere-Pau Vázquez 2024, Guozheng Li et. al 2024, Zhongzheng Xu et. al 2024) were all testing models with high parameter size, such as GPT3 or GPT4. Vázquez (2024) examined whether state-of-the-art LLMs are ready to perform visualization tasks in practical contexts, while Li et al. (2024) provided a broad evaluation of visualization generation with large language models. Similarly, Xu et al. (2024) explored the ability of large models to perform low-level analytic tasks on SVG-based visualizations. While these studies highlight the potential of large LLMs to handle complex visualization and reasoning tasks, they do not address the performance of smaller, resource-efficient models that are increasingly relevant for deployment in everyday systems.

Our contributions are as follows:

1. We introduce a generated benchmark dataset of user visualization descriptions paired with expected chart types, generated using GPT-4 as a teacher model.
2. We evaluate three small-parameter open-source models—LLaMA 3.2 3B, LLaMA 3.1 8B, and Deepseek 7B—on chart generation tasks with different prompting strategies.

3. We provide reasoning prompts and prompt engineering for improving chart generation quality for lightweight LLMs.

II. Related Works

The problem of creating data visualizations with LLMs has been attracting researchers for a long time. For example Pere-Pau Vázquez 2024, tested both ChatGPT3 and ChatGPT4 in creating a number of different charts. Their tests for the LLaMA2, Code LLaMA2 and Mixtral models in their 7B parameter versions showed a lot of hallucinations and errors, which shows that the older models were not suitable for these type of problems. They also tested 3 different python libraries and 24 different number of chart types. Their results showed that the larger models have great results for most types of charts, which 16 out of 24 were generated successfully for ChatGPT3 and 19 out of 24 for ChatGPT4.

Other researchers such as Guozheng Li et. al 2024 decided to use different techniques for generating charts. This paper showcased the ability of GPT-3.5 to generate Vega-Lite specifications that can be used to display visualizations. A lot of prompt techniques were used to test the model, such as Chain-of-Thought, Rationale Engineering, Problem Decomposition. But the most notable in the results was the few-shot prompting technique that showcased better results over the zero-shot prompting. Their evaluation metrics consisted of a combination of the ground truth visualization, and Vega-Lite specification. The results were not satisfactory, probably because the models did not understand the Vega-Lite specification and format.

The researchers in Zhongzheng Xu et. al 2024 tested GPT-4-turbo-preview for the generation of SVG code for data visualizations. The model was tested with zero-shot prompting for 10 different tasks such as, retrieving values, filtering, sorting, clustering, and computing derived metrics. The model displayed great results for clustering or retrieving labeled values, but struggled for tasks that required mathematical reasoning or computation.

The LLaMA 3 family of models was first introduced in the paper Aaron Grattafiori et. al 2024. The models supported coding, reasoning, and tool usage, making them perfect for testing them in chart generation techniques. The LLaMA 3.2 3B parameter model is mostly meant to be used in mobile devices or edge devices, but still has all the benefits of the higher parameter models in the same family.

The Deepseek R1 7B model was introduced Daya Guo et. al 2025 and was trained using pure reinforcement learning methods. The models became popular for his incredible reasoning capabilities that are comparable with state-of-the-art models such as OpenAI's o1 series models. This shows a great potential in the chart generation and understanding domain.

III. Overview of techniques

Since the goal is to test the model's ability to understand the user needs and generate a chart, a proper dataset and prompt techniques were needed. All techniques for chart generation, including the prompts, models, dataset and evaluation metrics are described in the following sections:

A. Selection of charts and Dataset

The dataset is a synthetically generated sample of user visualization descriptions and the expected chart type. The dataset consists of 30 descriptions, each generated with GPT 4 that is used as a teacher LLM. Since the models used are of smaller parameter size, a bigger model that has been trained on significantly higher data can be used as a baseline to create a testing dataset.

Most of the problems are simple, requesting the model to visualize simple relationships, or distributions in the dataset. The chart types used are: Histogram, Bar chart, Scatterplot, Pie chart and Line chart.

B. Selection of models

Three models in total were selected to be used in this research. The models were selected based on various characteristics such as parameter size, intended use, reasoning capabilities etc. The LLaMA models are run using Ollama on an RTX 4060 GPU, utilizing the Q4_K_S quantized versions, while the Deepseek model was used in its base form without any quantization.

- Llama 3.1 8b: A relatively lightweight model fine-tuned for tool calling, representing the latest in the LLaMA 8B family.
- Llama 3.2 3b: This model is intended for usage in mobile phones based on the small parameters and lightweight architecture, while still utilizing all characteristics of the higher parameter models in its family

- Deepseek 7b: One of the more popular open-source models, it demonstrates strong reasoning capabilities compared to other models such as LLaMA or OpenAI's o1 models

C. Chart Generation Techniques

Several chart generation techniques were used to test the models understanding of chart creation. For each model, a general system message was used to direct the model to respond in a structured manner. Using custom tags like <chart-code></chart-code> to extract the code snippets and chart types that the model generates. Some prompt engineering was also used to direct the model not to generate a sample dataset, but use the already loaded one.

In the user message, a sample of the dataset was given to the model as reference of the types and names of columns in the dataset. Besides the sample example, each user message was also altered based on the tasks that the models need do. This paper covers 3 chart generating tasks including:

- Asking the model to pick and generate a chart

The model is given a vague description of what needs to be visualized, not giving any indication of what the chart type is needed. This tests the model's ability to decide which chart is the best to visualize the problem.

- Asking the model to pick and generate a chart + Reasoning

The model is prompted to generate his thinking and then generate a chart. Prompt engineering was needed for the model to give structured output, so one shot prompting was used. The example given was a sample output of how the model should generate his results. This improved the models results by a significant amount.

- Giving the model chart type and asking to generate it

The model is given a description and the type of chart that needs to be generated. This gives the model less freedom to generate different charts, so that the model's chart code generating techniques are tested.

By structuring the evaluation in this way, we highlight not only the models' chart generation skills, but also the impact of prompt design and reasoning techniques on improving performance. A sample of one used prompt can be seen in Figure 1.

“system”:
You are a text-to-chart generating model. Always generate the charts using python and matplotlib.
Always use the dataset example given that is already loaded with pandas as variable *df*. Do not generate code for loading the dataset!
Do not hallucinate examples or generate sample data. Generate only one code. Respond only with the code.
Generate the chart type in the beginning between tags <*chart-type*></*chart-type*>.
Generate the code between tags <*chart-code*></*chart-code*>

“user”:
Show the relationship between sleep duration and quality of sleep

Example data:

Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep
1	Male	27	Software Engineer	6.1	6
2	Male	28	Doctor	6.2	6
3	Male	28	Doctor	6.2	6
4	Male	28	Sales Representative	5.9	4
5	Male	28	Sales Representative	5.9	4

Figure 1Example user and system prompt for task ‘Asking the model to pick and generate a chart’

D. Evaluation metrics

A simple evaluation metric was used for evaluating the models responses. Since models with small parameter size were used, a lot of the responses were not in the correct format and the extraction of code snippets was failing. Thus, the models were evaluated using the following evaluation approaches:

1. Hit vs. Miss Accuracy
   - A simple metric that determines whether the model generated a complete code snippet that compiles, and also if the generated type is matched with the expected chart type
   - The overall hit percentage is measured based on the number of hits / total number of chart descriptions
2. Manual Quality Scoring
   - In addition to type-level accuracy, a manual scoring techniques was used that determines if the model captured the chart description with another chart type
   - Each response was assigned a score with three-point scale:
     - **0** - The model failed to generate a correct chart, or no chart was displayed.

   - **1** - A chart was generated, but it was not complete or could have been better represented with an alternative chart type.
   - **2** - The generated chart matched the user's problem.

Only manual quality score was used for the chart generation with given chart type problem, since the hit score is based on the models chart type selection capability.

IV. Results

In the following section, the results are displayed along with a few remarks on how the model handled the structured output generation.

	LLaMA 3.2 3B		**LLaMA 3.1 8B**		**Deepseek 7B**	
	Hit vs. Miss	Manual Scoring	Hit vs. Miss	Manual Scoring	Hit vs. Miss	Manual Scoring
Ask generation	**0.3**	**0.5**	**0.63**	**0.8**	**0.6**	**0.48**
Ask generation + Reasoning	**0.76**	**0.82**	**0.9**	**0.92**	**0.57**	**0.67**
Give chart type	/	**0.95**	/	**0.96**	/	**0.58**

Figure 2Results from all tasks and used models

- Asking the model to pick and generate a chart
  - LLaMA 3.2 3B - The model usually chooses to generate the simplest chart based on it's knowledge, not trying to generate something that matches the users problem better. Because of this the Hit vs. Miss score is quite low. On the other side, based on the manual scoring, the model still gives a chart with a different type that still represents the users problem.
  - LLaMA 3.1 8B – This model gives quite an improvement over the smaller 3B model. The model generates the expected chart type and also has a high manual score. The model also makes less mistakes, with only one fail in the code generation part.
  - Deepseek 7b – While the model almost matches the Hit vs. Miss score of the LLaMA 3.1 model, the manual score is the lowest of the models. This is because the model mostly generates a response that does not display the chart, or generates code that does not compile.

- Asking the model to pick and generate a chart + Reasoning
  - LLaMA 3.2 3B – Big improvements were seen for this model with both scoring techniques. The model gave much more structured outputs that could be parsed and executed correctly. Also, the model generated more versatile charts that aligned with the expected one from the testing dataset.
  - LLaMA 3.1 8B – This model also saw a lot of improvement with the thought generation technique. The Hit vs. Miss metric saw a big improvement, most likely because the model now gives more structured outputs.
  - Deepseek 7b – On the opposite, this model performed with worse precision. This might show some bigger bias in the models training, that prevents it from giving structured outputs. The Manual score has increased, although not by a large margin.
- Giving the model the cart type
  - LLaMA 3.2 3B – With this technique, the model has displayed its chart generation techniques and now gives very high scores. This means that the model is much better in generating chart code, than determining the chart type from a description.
  - LLaMA 3.1 8B – This model follows the results from the previous with a small improvement, showing that parameter number does not matter for this type of problem.
  - Deepseek 7b – Following the model's results from the previous tasks, not much improvement was seen with the reasoning technique. This could point to a problem of generating chart code that the model may not be suited for.

V. Conclusion

This research experimented chart generation techniques with various LLM models with small number of parameters. While a lot of work was spent prompting the model to generate a structured output, the results show that careful prompt design and reasoning instructions significantly influence the performance.

The LLaMA models show high results when prompted with reasoning techniques, showcasing their capability to understand users needs and generate suitable charts based only on user visualization descriptions. The models also showed very high scores when only generating chart types and demonstrated that parameter size does not matter for this kind of task

On the other hand, the Deepseek model showed a lot of problems with these tasks. Even when prompted to give reasoning, the model struggled to generate a structured output and suitable charts from the given user descriptions. Giving the expected chart type also did not improve the models results and it could even be said that this restriction highlighted the model's weakness in generating chart code.

Therefore, even though chart generation is a task well suited for LLMs, proper model and prompt engineering is needed to get satisfying results.

VI. References

[1] Pere-Pau Vázquez 2024, "Are LLMs ready for Visualization?"

[2] Guozheng Li et. al 2024 "Visualization Generation with Large Language Models: An Evaluation"

[3] Zhongzheng Xu et. al 2024 "Exploring the Capability of LLMs in Performing Low-Level Visual Analytic Tasks on SVG Data Visualizations"

[4] Aaron Grattafiori et. al 2024 "The Llama 3 Herd of Models"

[5] Daya Guo et. al 2025 "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning"