Prompt Engineering and In Context Learning (ICL) Report

Matteo Gianvenuti

Contents

[Introduction 3](#_Toc198203046)

[1. Prompt structure 3](#_Toc198203047)

[1.1 The naming problem and the model dimension limit 4](#_Toc198203048)

[1.2 Textual representation 5](#_Toc198203049)

[1.3 JSON representation 6](#_Toc198203050)

[1.4 Tabular representation 7](#_Toc198203051)

[1.5 Questions examples 8](#_Toc198203052)

[2. Questions and answers dataset 9](#_Toc198203053)

[3. Models and results 9](#_Toc198203054)

[3.1 meta-llama/llama3.2-3B-Instruct 9](#_Toc198203055)

[3.2 microsoft/Phi-4-mini-reasoning 10](#_Toc198203056)

[3.3 microsoft/Phi-4-mini-instruct 11](#_Toc198203057)

[3.4 mistralai/Mistral-7B-Instruct-v0.3 11](#_Toc198203058)

[3.5 google/gemma-3-4b-it 11](#_Toc198203059)

[Figure 1 The sample system 3](#_Toc198203060)

[Figure 2 System interconnections in natural language 6](#_Toc198203061)

[Figure 3 JSON description of the Switch-1 7](#_Toc198203062)

[Figure 4 Interconnections' table of the system 8](#_Toc198203063)

[Figure 5 Questions and answers examples in the prompt 9](#_Toc198203064)

# Introduction

As result of the first kick-off meeting, the purpose of this report is to evaluate the ability of LLMs in understanding the hardware structure of a sample system and its interconnections. It is also to determine whether representation of the system architecture is more effective in making the model understand the system architecture.

I have experimented three representations: textual, JSON and tabular representation of the network architecture.

I have experimented with various models according to the available hardware capacity, especially meta-llama/llama3.2-3B-Instruct and microsoft/Phi-4-mini-reasoning.

A prompt, according to the representation, is feed to the models with the “system” role so it will be a permanent information during the questions. This prompt explains to the model its task, defines the system architecture, the naming rules and in which format the model should answer the questions.

The models have been tested on a set of questions about the network topology to verify the models’ understanding of the system architecture.



Figure 1 The sample system

# Prompt structure

The prompt is specific for each type of representation. At the beginning of the prompt is defined the model task:

“Your task:

You will be given a network topology. Your job is to understand the structure and answer the questions that follow.

Use the format shown in the examples to give your answers, do not add additional information.”

In the successive part initially, I put the components definitions as provided in the fifth slide of the first kick-off meeting:

“Definitions:

Console: A computer equipped with a monitor, keyboard, and mouse, allowing operators to interact with a system.

Switch: A networking device that connects multiple computers or devices, efficiently managing data traffic within a network.

Workstation: A high-performance computer.

SBC: (Single Board Computer) A compact computer with all essential components (CPU, memory, I/O) integrated onto a single board, used in embedded systems.

NAS: (Network Attached Storage) A dedicated storage device connected to a network, allowing multiple users to store and access files remotely.

Server: A powerful computer that provides services or resources to other computers, managing data, applications, or networks.”

After some experiments I removed it because it was not helpful in interconnections understanding and the size of the prompt matter.

In the successive part there are naming/critical rules that instruct the model on how components are called. Subsequently, are listed all the components.

## 1.1 The naming problem and the model dimension limit

The tests for this part have been done with meta-llama/llama3.2-3B-Instruct model and the textual representation.

In the first test I noticed most errors were caused by a similar nomenclature in multi-device components (e.g., Switch-1, Switch-2, Swtich-3). In questions like “What is connected to the Workstation-1-eth0?” the model answered more than one time like “The Switch-3-port-2” while the correct answer was “The Switch-2-port-2”. Then to better highlight the number that makes the devices different I tried a different representation by separating the device name and number from the port/interface with the colon symbol (‘:’). This did not change too much. So, I realized that the problem was due to the size of the model also because the model did not answer correctly to other questions (like “What is connected to ...?”) where the answer was clearly defined in the prompt where the topology is defined. Indeed, a smaller size model has less storage capacity. To demonstrate this initially, I expanded the context resulting in a worsening of accuracy from an average of 92% to 81% (check also the chapter 3).

In the following table the results (accuracy in answering) of the first tests with twenty questions. For all the other tests in the chapter 3 I used the expanded context with thirty questions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Device name format | Run 1 | Run 2 | Run 3 | Mean |
| Colon | 85.0% (17/20) | 70.0% (14/20) | 100.0% (20/20) | 85.0% |
| Dash | 90.0% (18/20) | 87.5% (17.5/20) | 100.0% (20/20) | 92.5% |

After I did the opposite experiment. So, I reduced the size of the system, only one switch with few connected components. This led the accuracy to an average of 92.94%.

I also feed the system architecture to larger models like GPT-4 and DeepSeek to verify it again. These models correctly understand and answer questions.

To calculate the accuracy in this case I defined a smaller dataset with 34 questions (to test all the interconnections of the smaller system). I feed to the model all the 34 questions for five runs. For simplicity I used only the “dash” representation for device names, also because as I said before it does not really matter. For this reason, I decided to test only the dash name format with the other models.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Mean |
| 88.2% (30/34) | 94.1% (32/34) | 91.2% (31/34) | 97.1% (33/34) | 94.1% (32/34) | 92.94% |

## 1.2 Textual representation

In this representation the system interconnections are described in natural language:

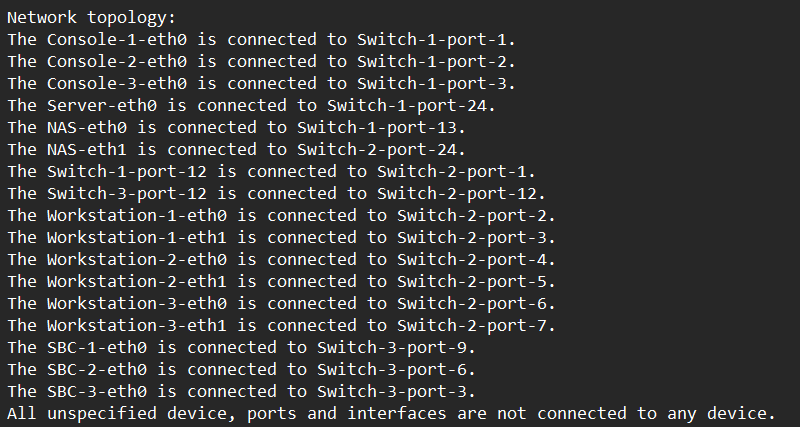


Figure 2 System interconnections in natural language

## 1.3 JSON representation

In this representation I described the network as JSON objects. I introduced this representation in the prompt as there:

“Interface connection rules:

If connected\_to is null, the interface is not connected.

If connected\_to has a value, it tells which device and interface it is connected to.

To find what is connected to something like SBC-1-eth2, search all devices to check if any interface points to: "device": "SBC-1" and "interface": "eth2"

Data format:

The network is described in JSON. At the top level, there is a "devices" list.

Each device has: a unique "name", a "type" (like Switch, Server, etc.) and an "interfaces" list.

Each interface has: a "name" (e.g., eth0, port-2) and a "connected\_to" field, which can be null or an object with "device" and "interface".”

Here an example for the Switch-1:

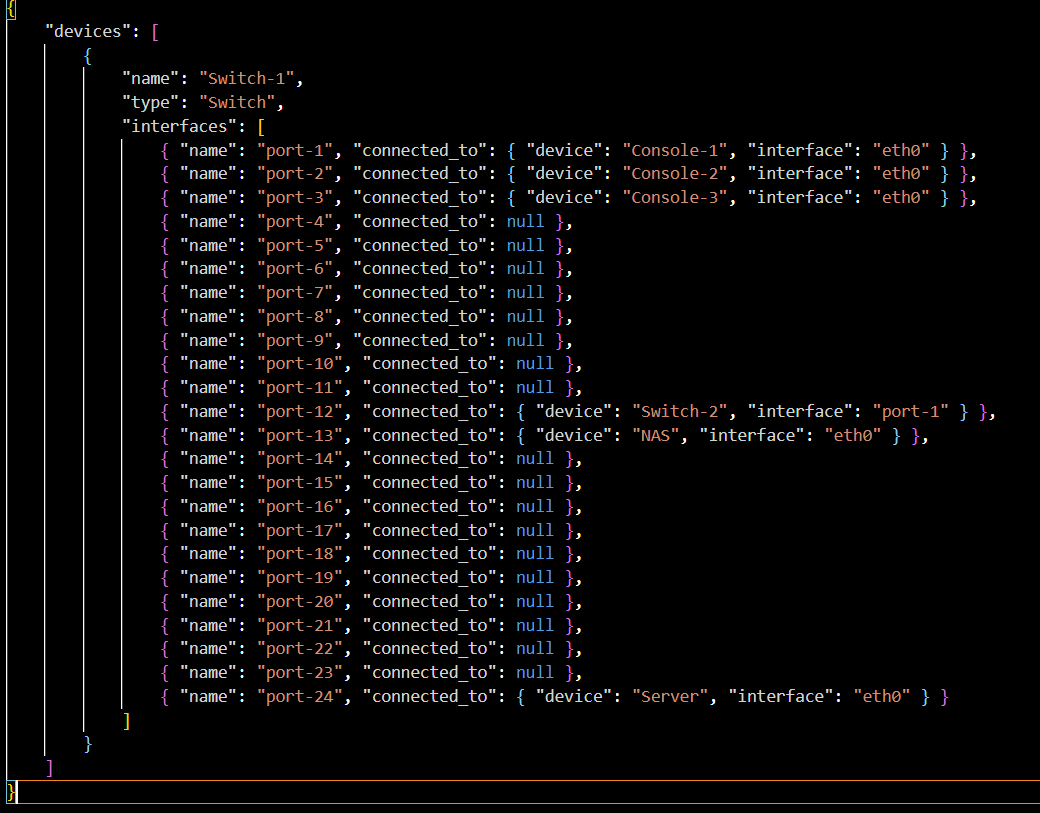


Figure 3 JSON description of the Switch-1

All the devices are listed inside the “devices” list object.

## 1.4 Tabular representation

The table is manually hardcoded:

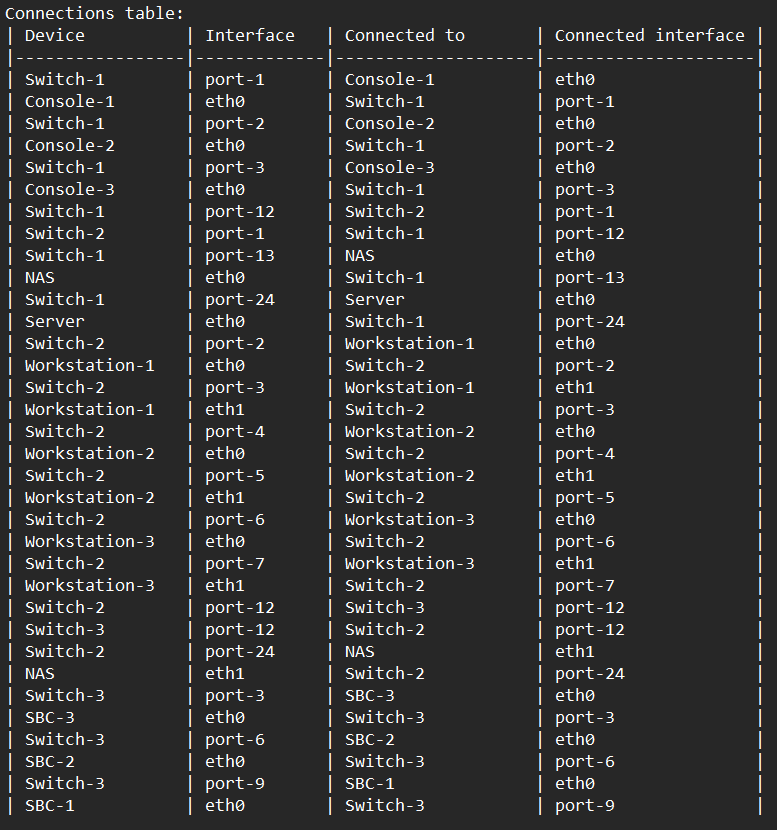


Figure 4 Interconnections' table of the system

## 1.5 Questions examples

In the last part of the prompt, I put some questions and answers as example. This teaches the model to respond in that format. At this point of the work, it is mainly useful to parse the answers and automatically evaluate the accuracy.

Not all models clearly understand that they must respond in the specified way. So, I added a parenthetical note to the prompt to clarify it, but models like mistralai/Mistral-7B-Instruct-v0.3 still don’t follow it, while meta-llama/llama3.2-3B-Instruct already understood it correctly from the start.

A computer screen shot of a black screen

AI-generated content may be incorrect.

Figure 5 Questions and answers examples in the prompt

# Questions and answers dataset

I built a dataset with 94 questions and corresponding answers about the system interconnections topology. The dataset contains questions like “What is connected to A?”. Questions are dual so the same connection may be asked two times in a different way, for example if A and B are connected you can ask “What is connected to A?” but also “What is connected to B?”. In the dataset there are also few questions about components that do not exists.

To calculate the accuracy, 30 questions are randomly extracted and asked to the model. Due to GPU memory limitations, I could not ask the models all questions at once.

I calculate the accuracy in reporting the asked questions and in correctly responding to questions. All the models can report the questions (the accuracy is always 100%), so they know the request but in answering/reasoning they have more difficulties.

# Models and results

## 3.1 meta-llama/llama3.2-3B-Instruct

I tested this model with 8-bit double quantization.

This model has been tested on all the representation. In few runs the model did not answer all the questions, so I ran it again in those cases.

In the following table the accuracy in answering the 30 questions with the textual representation:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Device name format | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Mean |
| Colon | 86.7% (26/30) | 83.3% (25/30) | 73.3% (22/30) | 73.3% (22/30) | 86.7% (26/30) | 80.66% |
| Dash | 80.0% (24/30) | 83.3% (25/30) | 73.3% (22/30) | 83.3% (25/30) | 90.0% (27/30) | 81.98% |

In the following table the accuracy in answering the 30 questions with the JSON representation:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Device name format | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Mean |
| Colon | 80.0% (24/30) | 93.3% (28/30) | 83.3 (25/30) | 80.0% (24/30) | 76.7% (23/30) | 82.66% |
| Dash | 63.3% (19/30) | 83.3% (25/30) | 83.3% (25/30) | 73.3% (22/30) | 90.0% (27/30) | 78.64% |

In the following table the accuracy in answering the 30 questions with the tabular representation:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Device name format | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Mean |
| Colon | 63.3% (19/30) | 70.0% (21/30) | 66.7% (20/30) | 80.0% (24/30) | 93.3% (28/30) | 74.6% |
| Dash | 86.7% (26/30) | 80.0% (24/30) | 50.0% (15/30) | 53.3% (16/30) | 63.3% (19/30) | 66.66% |

The tabular representation has the worst performances.

## 3.2 microsoft/Phi-4-mini-reasoning

I tested this model with 8-bit double quantization.

This model is focused on reasoning, it has the best performances, but it is a bit bigger (3.84B) than the used llama so after two test with 30 and one with 25 questions the hardware stopped working (always out of memory errors).

Since it is focused on reasoning was better able to understand logical relationships and answer correctly.

In this table I report the results of the few tests (textual representation):

|  |  |  |  |
| --- | --- | --- | --- |
| Run 1 | Run 2 | Run 3 | Mean 30q |
| 96.7%  (29/30) | 90.0%  (27/30) | 84.0%  (21/25) | 93.35% |

This model is 2x or 3x slower than the llama used. In most other cases it answered but only to 15-20 of 25-30 questions.

## 3.3 microsoft/Phi-4-mini-instruct

The hardware did not support this model with both 4-bit and 8-bit quantization.

## 3.4 mistralai/Mistral-7B-Instruct-v0.3

I tested this model with 4-bit double quantization, the hardware did not support the 8-bit quantization. I reduced also the number of questions to 25.

The model struggles to follow the Q&A output format, often ignoring the prompt and answering however it wants. I had to run it multiple times before getting the desired responses. Its poor performance is likely due to quantization or hardware limitations.

In the following table the answers accuracy with the textual representation:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Mean |
| 76.0% (19/25) | 64.0% (19/25) | 72.0% (18/25) | 72.0% (18/25) | 92.0% (23/25) | 75.2% |

## 3.5 google/gemma-3-4b-it

The hardware did not support this model with both 4-bit and 8-bit quantization.