

DIPARTIMENTO DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Magistrale Ingegneria Informatica

Enhancing Fault Isolation in Hardware Systems Using Large Language Models (LLM)

Miglioramento dell'isolamento dei guasti nei sistemi hardware utilizzando modelli linguistici di grandi dimensioni (LLM)

Relatore:

Prof. Gianfranco Lombardo

Tesi di Laurea di:

Matteo Gianvenuti Matricola: 364979

Table of Contents

[Introduction 2](#_Toc200014897)

[1. State Of The Art 3](#_Toc200014898)

[1.1 Large Language Models (LLMs) 3](#_Toc200014899)

[1.2 Self-supervised learning (SSL) 3](#_Toc200014900)

[1.3 Prompt Engineering 3](#_Toc200014901)

[1.4 Generative pre-trained transformer (GPT) 5](#_Toc200014902)

[1.5 Llama 5](#_Toc200014903)

[1.6 Diagnostics 5](#_Toc200014904)

[Bibliography 5](#_Toc200014905)

Introduzione

- Ambito dell’attività di tesi

- Obiettivi dell’attività di tesi

Stato dell’arte

- Documentare tutto ciò che esisteva prima della tesi in termini di tecnologie, metodologie e modelli

2/3 capitoli di metodologia in cui spiega cosa ha effettivamente svolto, progettato e realizzato (architettura software, addestramento di modelli, dati utilizzati e loro composizione etc..)

- 1 capitolo su i risultati

- 1 breve capitolo di conclusioni in cui ricapitola il tutto traendo le conclusioni e proponendo degli sviluppi futuri.

***Enhancing Fault Isolation in Hardware Systems Using Large Language Models (LLM): A Real-Time Diagnostic Framework***

*This thesis explores the application of Large Language Models (LLM) to improve fault isolation in hardware systems. Thanks to their ability to analyse large amounts of unstructured data, LLMs can identify patterns and anomalies in system logs and telemetry data collected from hardware devices. The objective is to develop a framework that uses LLMs to diagnose faults in real-time, reducing downtime and improving the overall reliability of the system. Case studies and experimental results will be presented to demonstrate the effectiveness of the proposed approach.*

# Introduction

In this thesis work we want to explore the use of Large Language Models (LLMs) in hardware diagnostics. LLMs have shown good memorization and "understanding" capabilities of natural language, even in reasoning tasks. Especially in the case in which the models know the environment in which they operate.

In our case, we initially want to make the LLMs understand how a hardware architecture is structured so that the model is able to solve the diagnostics at its best. Subsequently, we want to exploit the capabilities of the LLMs to identify any problems in the hardware by analysing system logs, telemetry data and hardware functionality test results. To carry out these activities it was necessary to create a specific dataset and a simulation environment. We also implemented the Open WebUI graphical interface.

So, we have two objectives, the first is to verify that the LLMs are able to understand the hardware architecture. While the second is to verify that the LLMs are able to understand and solve hardware problems. This work was based on In-Context Learning (ICL) and Prompt Engineering. In particular, we compared the performance of different models according to various metrics.

# 1. State Of The Art

## 1.1 Large Language Models (LLMs)

LLMs are machine learning models with many parameters, designed for natural language processing tasks, especially text generation. These models are typically built using deep neural networks, particularly transformer architectures, and are trained on massive corpora of textual data. They are trained with self-supervised learning on a vast amount of text. Their strength lies in the ability to capture complex syntactic, semantic, and contextual patterns, allowing them to perform a wide range of tasks such as translation, summarization, question answering, and reasoning.

LLMs can be fine-tuned for specific tasks or guided by prompt engineering, as done in this work. These models acquire predictive power regarding syntax, semantics, and ontologies inherent in human language corpora, but they also inherit inaccuracies and biases present in the data they are trained in.

The core idea behind LLMs is to learn a statistical model of language by predicting the next word (or token) in a sentence, given a context. This is achieved using self-supervised learning, where the model learns from raw text without the need for labelled data. During training, the model builds internal representations that encode meaning, structure, and dependencies across sequences of words.

The largest and most capable LLMs are generative pre-trained transformers (GPTs).

Modern LLMs like OpenAI’s GPT, DeepSeek’s DeepSeek series or Meta’s LLaMA models are characterized by having billions or even trillions of parameters. The performance of these models tends to improve significantly as the number of parameters and the size of the training data increase, a phenomenon often referred to as “scaling laws.”

In addition to linguistic capabilities, recent LLMs demonstrate emergent properties such as reasoning, commonsense inference, and even basic code generation. However, they also exhibit limitations, including sensitivity to prompt phrasing, hallucination of facts, and the presence of biases derived from the training data. So, in the context of training LLMs, datasets are typically cleaned by removing low-quality, duplicated, or toxic data. Cleaned datasets can increase training efficiency and lead to improved downstream performance.

As all the others machine learning (ML) algorithms, LLMs process numbers rather than text, the text must be converted to numbers through the tokenization process. Tokenization is a fundamental process in LLM pipelines. Before text is processed by a model, it is converted into tokens. In the first step, a vocabulary is decided upon, then integer indices are arbitrarily but uniquely assigned to each vocabulary entry, and finally, an embedding is associated to the integer index.

In our thesis work, LLMs serve a dual role: first, as systems capable of understanding the architecture and components of hardware environments through descriptive prompts; second, as tools capable of interpreting system logs and telemetry to support fault diagnosis.

## 1.2 Self-supervised learning (SSL)

SSL is a paradigm in machine learning where a model is trained on a task using the data itself to generate supervisory signals, rather than relying on externally provided labels. In the context of neural networks, self-supervised learning aims to leverage inherent structures or relationships within the input data to create meaningful training signals. SSL tasks are designed so that solving them requires capturing essential features or relationships in the data.

Auto-associative self-supervised learning is a specific category of SSL where a neural network is trained to reproduce or reconstruct its own input data. In other words, the model is tasked with learning a representation of the data that captures its essential features or structure, allowing it to regenerate the original input.

## 1.3 Prompt Engineering

Prompt engineering is the process of structuring or crafting an instruction in order to produce the best possible output from a generative artificial intelligence (AI) model.

In case of text-to-text language models a *prompt* is natural language text describing the task that an AI should perform. It can be a query, a command, or a longer statement including context, instructions, and conversation history. Prompt engineering may involve phrasing a query, specifying a style, choice of words and grammar, providing relevant context, or describing a character for the AI to mimic.

There are many different types of prompt engineering:

* Chain-of-Thought (CoT): it is a reasoning technique that goes through several intermediate steps before giving a definitive answer to a question.

The CoT prompting improves reasoning ability by inducing the model to answer a multi-step problem with steps of reasoning that mimic a train of thought. These techniques were developed to aid common sense reasoning and multi-step reasoning.

The prompt specifies some examples of questions and answers (Q&A); this improved the results considerably. Even just writing "let's think step by step" improved the results a lot. Allowing you to generalize and avoid writing many specific examples that consume the context.

* In-Context Learning (ICL): it is the ability of models to learn information temporarily. The information of interest is specified in the prompt, typically in the initial part. This information is then used by the model in the session to answer questions by generating text, this allows the model to be trained without necessarily having to perform fine-tuning.

The weights of the network are not updated, so the information is temporary. Starting a new instance the model does not retain this knowledge.

* Prompting to estimate model sensitivity: research consistently demonstrates that LLMs are highly sensitive to subtle variations in prompt formatting, structure, and linguistic properties. Linguistic features significantly influence prompt effectiveness—such as morphology, syntax, and lexico-semantic changes—which meaningfully enhance task performance across a variety of tasks.

To address sensitivity of models and make them more robust, several methods have been proposed. Format Spread facilitates systematic analysis by evaluating a range of plausible prompt formats, offering a more comprehensive performance interval. Similarly, Prompt Eval estimates performance distributions across different prompts, enabling robust metrics such as performance quantiles and accurate evaluations under constrained budgets.

* Retrieval-augmented generation (RAG): it is a technique that enables generative AI models to retrieve and incorporate new information. It modifies interactions with a LLM so that the model responds to user queries with reference to a specified set of documents, using this information to supplement information from its pre-existing training data. This allows LLMs to use domain-specific and/or updated information.
* Few-shot: the model is already pre-trained and fine-tuned to become instruct. Few-shot means that in inference in the prompt are put from zero to 4/5 task examples before the request for an actual task.

There are many other techniques, but they are not relevant to this work. In this work we mainly focused on ICL and CoT, building a customized few-shot technique.

## 1.4 Generative pre-trained transformer (GPT)

## 1.5 Llama

## 1.6 Diagnostics

# Bibliography

[] LLMs. (Online) <https://en.wikipedia.org/wiki/Large_language_model>.

[https://web.stanford.edu/~jurafsky/slp3/](https://eur01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fweb.stanford.edu%2F~jurafsky%2Fslp3%2F&data=05%7C02%7Cmatteo.gianvenuti%40studenti.unipr.it%7C3eb6fbcc055f4e8a0aff08dd8bd29bee%7Cbb064bc5b7a841ecbabed7beb3faeb1c%7C0%7C0%7C638820458860462928%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=R76CP%2Fvpd1I40a8MReCrxBrrjUE6ColECYnknFGpswY%3D&reserved=0) book various arguments (it should cover almost all)

[] Prompt engineering (Online) <https://en.wikipedia.org/wiki/Prompt_engineering>.

[]