**Presentation speech in Italian**

Matteo Gianvenuti

**Slide 1**

Buongiorno a tutti, presento la tesi Enhancing Fault Isolation in Hardware Systems Using Large Language Models (LLMs).

**Slide 2**

Gli obbiettivi della tesi sono due:

1. Sperimentare l’utilizzo degli LLMs nella diagnostica hardware, e quindi verificarne la capacità di individuare anomalie e ambiziosamente le loro possibili cause.
2. Utilizzare risorse hardware limitate come quelle che si possono avere su un portatile per via del contesto aziendale.

**Slide 3**

Per raggiungere gli obbiettivi ho diviso il lavoro in due step principali. Il *primo* consiste nel verificare la capacità degli LLMs di comprendere un’architettura di sistema, la sua topologia e le sue interconnessioni. Individuando la rappresentazione più adatta. Il *secondo* andare effettivamente a individuare anomalie e possibili cause tramite uso di LLM istruito con In-Context Learning e Prompt Engineering.

**Slide 4**

Per raggiungere gli obbiettivi, è stato necessario realizzare un ambiente di simulazione descritto dallo schema in figura. Ciò permette la sperimentazione senza necessita di un sistema hardware reale.

**Slide 5**

Lo step uno è composto da tre attività principali.

La *prima* è l’analisi dello stato dell’arte per individuare le attuali soluzioni, che si concentrano solo sulla rilevazione delle anomalie e la loro semplice classificazione anche tramite LLMs.

La *seconda* è l’istruire gli LLMs per consentirgli di comprendere l’architettura e topologia del sistema.

Mentre la *terza* è verificare le capacità degli LLMs di comprensione del sistema.

**Slide 6**

Per l’attività di prompt engineering del primo step ho realizzato un prompt dove all’inizio descrivo il task che il modello deve svolgere, successivamente lo istruisco sulle componenti del sistema e sulla topologia delle connessioni, e infine gli mostro degli esempi per insegnargli a rispondere nel modo desiderato.

L’esempio nella figura in basso evidenzia la capacità di generalizzazione degli LLMs.

**Slide 7**

Per la rappresentazione della topologia delle connessioni del sistema ho sperimentato tre formati: testuale, json e tabellare.

**Slide 8**

Qui ho riporta l’accuratezza nel comprendere l’architettura del sistema e le sue connessioni. Il modello mediamente migliore è Llama. È anche il più veloce nel rispondere. Invece Phi ha avuto prestazioni molto alte ma essendo un modello di ragionamento è più lento e l’hardware ad un certo punto non lo ha più supportato. Mentre Mistral a avuta performance nettamente più basse degli altri due eccetto in una run (Per avere l’evidenza statistica ho svolto più run, ovviamente). La rappresentazione migliore sono quella testuale e json più o meno a parimerito a meno di un uno percento.

**Slide 9**

Inoltre, è risultato che un modello più grande o un sistema più piccolo portano a risultati migliori. E quindi una maggiore comprensione dell’architettura del sistema. Per dimostrarlo ho sperimentato Llama su un sistema più piccolo. Ciò ha portato ad una accuratezza media superiore al 90% mentre nel sistema originario era mediamente superiore all’80%. Questo per capire i limiti del modello.

**Slide 10**

Per lo step due è stato necessario creare anche degli scenari in cui identificare possibili anomalie e cause. In particolare, sono stati sviluppati tre scenari: uno dove tutto funziona correttamente, un secondo dove lo switch due è guasto ed un terzo dove la SBC2 è guasta.

Per ognuno di questi scenari sono stati eseguiti i test dello strumento di diagnostica aziendale per realizzare un dataset su cui preparare e sperimentare il modello.

**Slide 11**

Nello step due ho svolto sperimentazioni sia con few-shot che senza. Per le prime sperimentazioni senza few-shot, ho fornito al modello questo prompt dove lo istruisco sul suo task e gli spiego come sono strutturati i test, e come interpretare i risultati.

**Slide 12**

In questa prima sperimentazione solamente nel primo caso i modello ha correttamente individuato le anomalie (in questo caso l’assenza di anomalie) mentre negli altri non sono state individuate le anomalie e le cause.

**Slide 13**

Per le sperimentazioni few-shot, avendo tre scenari a disposizione, ho svolto sperimentazioni in cui a rotazione una volta uno (o due nel caso di two-shot) scenario è utilizzato come esempio per istruire il modello mentre gli altri o l’altro scenario sono utilizzati come domande per misurare le prestazioni.

Quindi in questi casi c’è una prima parte del prompt dove spiego il task ed i test. Una seconda parte dove metto i test di esempio ed infine una terza dove metto la deduzione logica di esempio che il modello deve svolgere sui test.

Qui a destra ho riportato l’inferenza di esempio che il modello deve svolgere, una per ogni scenario.

**Slide 14**

Nella maggior parte dei casi il modello ha correttamente individuato le anomalie. Ha anche individuale alcune possibili cause, secondo lui, ma non ha trovato le cause reali degli scenari, questo può essere dovuto al fatto che ci sono troppi test e troppi dati.

**Slide 15**

Come conseguenza ho ridotto il numero dei test e preparato il modello tramite il prompt per queste situazioni ovvero l’ho istruito solo sul tipo di test che deve comprendere. Di conseguenza il prompt è più piccolo.

Ho sperimento due situazioni una prima dove vengono mostrati e viene chiesto al modello solo di analizzare test riguardanti le connessioni ed il loro stato e velocità delle porte.

Ed una seconda dove ci si restringe solamente a test di tipo PBIT (Power-on Built-in Test sono test che verificano durante il processo di boot la salute della macchina e delle sue funzionalità). Per la seconda situazione ho anche utilizzato un sistema lievemente più piccolo.

**Slide 16**

Nella prima situazione il modello ha individuato la presenza o assenza di anomalie in due casi su tre e le cause di anomalia in un caso su due (perché nello scenario uno non ci sono anomalie e quindi non ci sono cause).

Mentre nel caso dei test di tipo PBIT sono state le anomalie o la loro assenza in tutti i casi

**Slide 17**

In conclusione, quando il modello ha abbastanza conoscenza sulla situazione specifica e non ha troppe informazioni da elaborare, quindi un contesto o un input non troppo grande, riesce ad individuare le anomalie e a volte anche le cause corrette (corrette perché il modello propone quasi sempre delle possibili cause). Ciò può rendere il processo di diagnostica più veloce.

**Slide 18**