Università degli Studi di Salerno

Dipartimento di Informatica

Corso di Ingegneria dei Linguaggi di Programmazione

Grammo

Implementazione del compilatore per il linguaggio di programmazione Grammo

Salvatore Di Martino

NF22500114

Utilizzo dei Large Language Models (LLM) nel progetto Grammo

Strumenti e modelli utilizzati

Nel corso dello sviluppo del progetto sono stati utilizzati principalmente due LLMs: **Gemini 3 Pro (High)** e **ChatGPT 5.2 (modalità Auto/Thinking)**.
Gemini è stato impiegato tramite **Antigravity**, un ambiente di sviluppo integrato (IDE) basato su AI, all'interno del quale è stato realizzato gran parte del progetto. ChatGPT, invece, è stato utilizzato tramite interfaccia web, mediante il mio account personale Plus.

Ambiti e modalità di utilizzo

Gemini, integrato in Antigravity, è stato utilizzato prevalentemente per la **generazione di codice** e per l'ottenimento di **informazioni tecniche specifiche** relative al progetto, come dipendenze, istruzioni per l'esecuzione da linea di comando (CLI) e dettagli sulle fasi di analisi semantica e di generazione del codice (code generation).
La produzione del codice è sempre stata guidata da prompt scritti manualmente, nei quali ho descritto in modo esplicito la struttura, il comportamento e i requisiti delle componenti da implementare. In alcuni casi ho fornito anche esempi concreti, al fine di ottenere risultati più aderenti alle mie esigenze progettuali. Le informazioni richieste a Gemini sono state inoltre utilizzate per migliorare la comprensione del codice prodotto e come base per la successiva stesura della documentazione.

ChatGPT è stato impiegato principalmente per la **formattazione e la redazione della documentazione**, incluso il file README.md, partendo sia dalle informazioni tecniche ottenute tramite Gemini sia da testi "grezzi" scritti da me. È stato inoltre utilizzato per la **definizione della grammatica del linguaggio**, a partire da una mia descrizione iniziale semi-formale, e per la produzione della specifica in formato Lark. In questo ambito, ChatGPT si è dimostrato particolarmente accurato e affidabile.
Anche i file di test del linguaggio (*grammo_all_construct.gm, fibonacci.gm, calculator.gm e factorial.gm*) sono stati generati e verificati con l'ausilio di ChatGPT, che si è rivelato efficace nel riconoscimento dei costrutti sintattici e nella produzione di codice sintatticamente corretto. Infine, ChatGPT è stato utilizzato per effettuare **verifiche incrociate**, individuando difetti o possibili miglioramenti nel codice generato da Gemini. Questo mi ha permesso di fornire ad Antigravity indicazioni precise sulle modifiche da applicare.

Vantaggi e svantaggi riscontrati

I principali vantaggi di Gemini sono legati alla sua **integrazione nativa nell'IDE Antigravity** e alla capacità di generare grandi quantità di codice sfruttando un contesto molto esteso. In particolare, la modalità *Plan* di Antigravity si è rivelata estremamente utile, poiché consente di pianificare le operazioni, mostrando in anticipo il piano di implementazione e i task previsti. Questo approccio permette allo sviluppatore di avere una visione complessiva delle attività e di accettare o modificare il piano prima dell'esecuzione.
Un ulteriore punto di forza è la possibilità di eseguire automaticamente i comandi (previa autorizzazione dell'utente) e di verificarne l'output, fornendo feedback immediati sui risultati

dei test e sull'esecuzione del programma. Risulta inoltre molto efficace la visualizzazione delle modifiche al codice (*diff*), che consente di applicarle in modo controllato, file per file.

Tra gli aspetti negativi, si segnala la presenza iniziale di commenti eccessivamente legati al ragionamento interno del modello, poco adatti a un progetto accademico o professionale; tali commenti sono stati successivamente rivisti per renderli più chiari e uniformi. In alcuni casi il modello ha inoltre mostrato fenomeni di *hallucination*, omettendo istruzioni (commentandole) o introducendo piccoli difetti non critici che non riflettevano fedelmente il piano di implementazione. Questi problemi sono stati individuati anche grazie al supporto di ChatGPT. In rare occasioni, infine, Antigravity ha eseguito comandi che hanno causato il blocco dell'applicazione.

ChatGPT si è invece confermato particolarmente efficace per **analisi testuale, ragionamento e comprensione di richieste specifiche**, oltre che per la chiarezza e la qualità dell'output prodotto. Poiché è stato utilizzato prevalentemente per compiti meno critici dal punto di vista esecutivo, non sono emersi svantaggi rilevanti, e il modello ha sempre soddisfatto le mie esigenze.

Considerazioni finali e lezioni apprese

Ritengo che strumenti di questo tipo, e in particolare Antigravity, abbiano un impatto significativo sull'approccio di uno studente o di un programmatore allo sviluppo software e alla risoluzione dei problemi. Essi permettono di accelerare notevolmente il processo di sviluppo e di affrontare progetti più ambiziosi rispetto a quelli che si affronterebbero con strumenti tradizionali.
Tuttavia, è fondamentale utilizzarli con cautela: l'esperienza maturata mostra chiaramente come tali sistemi siano ancora soggetti a imprecisioni e allucinazioni, soprattutto in presenza di prompt poco chiari o incompleti. È quindi essenziale che chi li utilizza sia sempre in grado di comprendere e valutare criticamente il codice e i testi generati. Solo mantenendo questo controllo consapevole è possibile sfruttare appieno il potenziale offerto da questi strumenti senza comprometterne l'affidabilità.