



Dipartimento di Ingegneria Civile, Informatica
e delle Tecnologie Aeronautiche

Corso di Laurea in Ingegneria Informatica

Potenziare i chatbot basati su Large
Language Models attraverso Model
Context Protocol

Anno Accademico 2024/2025

Laureando

Del Prete Andrea

Matricola 589453

Relatore

Prof. Merialdo Paolo

Tutor

Dott. Di Nardo Giorgio

Ringraziamenti

Vorrei sinceramente esprimere gratitudine ai miei docenti, per la dedizione e la passione con cui mi hanno trasmesso le loro conoscenze, contribuendo in modo fondamentale alla mia crescita accademica; ai colleghi di Proge-Software, per la professionalità e la disponibilità con la quale mi hanno accolto, guidato e supportato durante il mio percorso di tirocinio, permettendomi di mettere in pratica quanto appreso e di acquisire nuove competenze; ai miei amici e colleghi di università, per il costante sostegno, la collaborazione e la condivisione di momenti indimenticabili, che hanno reso questo percorso più ricco e stimolante; ed infine alla mia famiglia, per il supporto incondizionato che mi hanno sempre dimostrato e senza la quale nulla di tutto questo sarebbe stato possibile.

Indice

Indice	2
Introduzione	6
1 Large Language Model - Cosa sono e come funzionano	8
1.1 Introduzione	9
1.2 Funzionamento	10
1.2.1 Panoramica	10
1.2.2 Tokenizzazione e pre-processing dei dati	10
1.2.3 Il Transformer: principio e componenti fondamentali	11
1.2.4 Stadi di adattamento: dal pre-training all'uso	11
1.3 <i>Pre-training</i>	12
1.3.1 Obiettivi di pre-training	12
1.3.2 Dataset di Addestramento	12
1.3.3 Sfide del Pre-training	13
1.4 Fine-tuning e Adattamento	14
1.4.1 <i>Supervised Fine-tuning</i>	14
1.4.2 <i>Instruction Tuning</i>	14
1.4.3 <i>Reinforcement Learning from Human Feedback</i>	15
1.4.4 <i>Domain Adaptation</i>	15
1.4.5 <i>Alignment Tuning</i>	15
1.5 Tecniche di Prompting e In-Context Learning	16

1.5.1	Zero-shot Prompting	16
1.5.2	Few-shot Prompting e In-Context Learning	16
1.5.3	Prompt Engineering	16
1.5.4	Prompting per il Ragionamento	17
1.5.5	Multi-turn Prompting	17
1.5.6	Tecniche Avanzate di Prompting	17
1.6	Applicazioni	19
1.6.1	Generazione e Comprensione del Linguaggio Naturale	19
1.6.2	Programmazione e Ingegneria del Software	19
1.6.3	Supporto alla Ricerca e alla Scienza	20
1.6.4	Medicina e Assistenza Sanitaria	20
1.6.5	Educazione e Apprendimento	20
1.6.6	Business e Produttività	21
1.6.7	Applicazioni Multimodali	21
1.6.8	Agenti Autonomi e Tool Use	21
1.7	Sfide, Limitazioni e Aspetti Etici	22
1.7.1	Allucinazioni e Aderenza ai Fatti	22
1.7.2	Bias ed Equità	22
1.7.3	Sicurezza e Abusi Potenziali	22
1.7.4	Interpretabilità e Trasparenza	23
1.7.5	Sostenibilità e Impatto Ambientale	23
1.7.6	Accessibilità e Disuguaglianze	23
1.7.7	Allineamento con Valori Umani	24
1.7.8	Aspetti Legali e Regolamentazione	24
1.8	Prospettive Future e Direzioni di Ricerca	25
1.8.1	Efficienza e Sostenibilità	25
1.8.2	Modelli Multimodali	25
1.8.3	Memory-Augmented Large Language Models	26
1.8.4	Autonomia e Agenti Intelligenti	26

1.8.5	Interpretabilità e Controllo	26
1.8.6	Sicurezza e Robustezza	26
1.8.7	Allineamento e Governance	27
1.8.8	Verso l'Intelligenza Artificiale Generale	27
2	Model Context Protocol	28
2.1	Funzionamento	28
2.1.1	Partecipanti	29
2.1.2	Livelli	29
	Livello dati	30
	Livello di trasporto	30
2.1.3	Protocollo del livello dati	31
	Primitive	31
	Notifiche	32
3	Ticket Management System	33
3.0.1	Scelta del Large Language Model	33
3.1	Struttura del progetto	34
3.1.1	TM.Shared	34
3.1.2	TM.Data	35
3.1.3	TM.CQRS	35
3.1.4	TM.Function	35
3.1.5	TM.Client	35
3.2	Il Model Context Protocol in pratica	36
3.3	Verifica pratica	38
3.4	Limitazioni	43
	Conclusioni e sviluppi futuri	44
	3FS [©] - Panoramica del funzionamento	44
	Ticket Management System applicato a 3FS [©]	45

Vantaggi attesi	45
Conclusione	46
Bibliografia	47

Introduzione

Siamo oramai in un'era in cui l'intelligenza artificiale è presente in ogni aspetto della nostra vita digitale, ancora di più dopo l'uscita, a fine 2022, di ChatGPT, che stabilì il record di applicazione con la più rapida crescita nella storia di Internet, ben 100 milioni di utenti in soli 2 mesi [1]. La sua diffusione fece crescere l'interesse per l'intelligenza artificiale generativa capace, cioè, di generare testi, audio, immagini o video a partire da un semplice "prompt" dell'utente.

Di particolare interesse sono le IA capaci di generare testo, i cosiddetti *Large Language Models* (LLM), letteralmente modelli linguistici di grandi dimensioni, in grado di emulare con sorprendente accuratezza le capacità conversazionali di un essere umano. Un'interessante conseguenza del modo in cui questi algoritmi sono stati creati, ma soprattutto della mole di dati usata per addestrarli, è stato l'emergere di comportamenti che potremmo considerare intelligenti, con un livello quasi pari a quello di un essere umano. Ad esempio, gli LLM sono in grado di ricordare informazioni, generare risposte articolate basate su un ampio contesto e ragionare su problemi complessi derivanti dalle branche più disparate, dalla meccanica quantistica alla biologia evolutiva.

Negli ultimi anni, è nato il desiderio di rendere ancora più sofisticate le abilità di questi modelli linguistici, finora relegate al "semplice" rispondere ai messaggi degli utenti. Per questo motivo Anthropic, una delle maggiori aziende nel settore della ricerca sugli LLM e creatrice dei molto diffusi modelli Claude, ha ideato un meccanismo che consente di raggiungere questo obiettivo: il Model Context Protocol (MCP). Questo protocollo open-source definisce una procedura standard che con-

sente alle applicazioni di fornire contesto ai modelli linguistici, permettendo loro di accedere a una serie di strumenti esterni e interagire con essi in modo sicuro e controllato. Grazie a ciò, gli LLM possono eseguire operazioni specifiche, recuperare informazioni o integrare funzionalità aggiuntive, migliorando l'efficacia e la personalizzazione delle risposte generate [2].

La seguente tesi sarà così strutturata:

- **Capitolo 1:** discussione nel dettaglio degli LLM, dalla loro struttura e funzionamento alle strategie che vengono usate dai ricercatori per renderli così sofisticati, oltre alle limitazioni di questi algoritmi e le sfide derivanti dalla loro diffusione.
- **Capitolo 2:** introduzione al Model Context Protocol di Anthropic.
- **Capitolo 3:** discussione sul progetto centrale di questa tesi, il *Ticket Management System*, volto a dimostrare nella pratica le funzionalità del Model Context Protocol. Il progetto in questione è un sistema software che consente agli utenti di interfacciarsi con un chatbot, rendendo più naturale la creazione di ticket che descrivono i problemi riscontrati e l'inoltro degli stessi a degli sviluppatori che possono visionarli e avviare delle procedure per risolverli. Verrà inclusa una conversazione, a titolo d'esempio, tratta dall'interazione con il chatbot, per dimostrare il corretto funzionamento del sistema.

Infine, seguirà una discussione sulle limitazioni del progetto e sulle prospettive future. In particolare, l'integrazione del *Ticket Management System* nel sistema software 3FS[©].

1

Large Language Model - Cosa sono e come funzionano

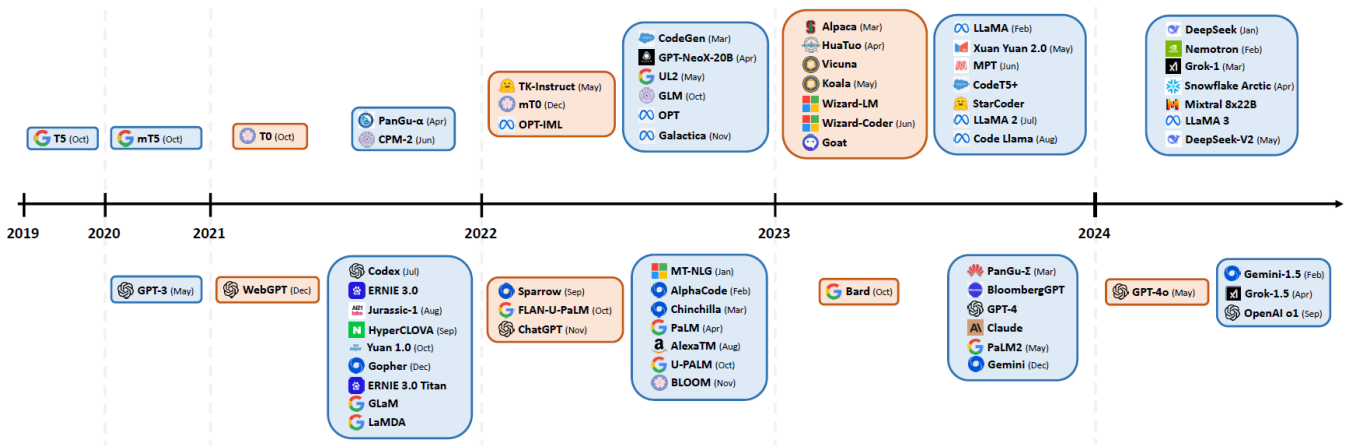


Figura 1: Cronologia dei principali *Large Language Models* (LLM) sviluppati fino al 2024. Sopra la linea ci sono i modelli open-source, mentre sotto ci sono quelli closed-source. [3]

1.1 Introduzione

Il linguaggio rappresenta uno strumento fondamentale per gli esseri umani, sia come mezzo di comunicazione e trasmissione di conoscenza, sia come interfaccia per interagire con le macchine. Lo sviluppo di modelli linguistici sempre più potenti è stato quindi al centro della ricerca in *Natural Language Processing* (NLP).

Storicamente, si è passati da approcci statistici (basati su n-grammi e modelli probabilistici) a modelli neurali, con una struttura ispirata a quella del cervello, fino ai modelli linguistici pre-addestrati (*Pre-trained Language Models*, PLM). Questi ultimi, addestrati su grandi corpus testuali, hanno introdotto il concetto di rappresentazioni generali del linguaggio, successivamente adattabili tramite *fine-tuning* a specifici compiti [4][5][6].

Un punto di svolta è stata l'introduzione dei ***Transformer*** [7], che hanno reso possibile scalare i modelli a miliardi di parametri grazie al meccanismo di attenzione, alla parallelizzazione e alla disponibilità di hardware e dataset sempre più estesi. Questo ha portato alla nascita dei **Large Language Models** (LLM).

1.2 Funzionamento

Illustriamo ora i concetti fondamentali per comprendere la struttura e il funzionamento dei *Large Language Models* (LLM).

1.2.1 Panoramica

Il testo dell'utente, il cosiddetto *prompt*, viene prima pre-processato e ridotto in *token*, ossia frammenti più piccoli, i quali vengono poi trasformati in rappresentazioni numeriche, oggetti matematici chiamati *embeddings*, che attraversano una rete neurale particolare chiamata Transformer, basata su meccanismi di attenzione, ottenendo come risultato un singolo token. Il testo iniziale, con l'aggiunta del token generato, viene poi dato in input all'LLM per generare un nuovo token e così via, generando infine una risposta al prompt dell'utente. Come ultimo passaggio, il modello viene addestrato, cioè i suoi parametri vengono automaticamente modificati, affinché le sue risposte siano sensate e coerenti.

Le diverse scelte in ciascuna di queste fasi, tokenizzazione, qualità dei dati, obiettivo di training e architettura, determinano in gran parte le capacità e i limiti del modello.

1.2.2 Tokenizzazione e pre-processing dei dati

La tokenizzazione [8] spezza il testo in unità elementari chiamate token: possono essere singoli caratteri, sottoparole [9], simboli [10] o parole complete. Prima della tokenizzazione si eseguono operazioni di pre-processing: filtraggio della qualità, deduplicazione e rimozione di informazioni sensibili. Inoltre, un buon pre-processing dei dati riduce: rumore, bias e contenuti non desiderati.

1. Large Language Model - Cosa sono e come funzionano

1.2.3 Il Transformer: principio e componenti fondamentali

Il Transformer è l'architettura alla base degli LLM moderni. Si basa su un meccanismo chiamato *self-attention* [11], attraverso il quale, nella trasformazione dei token in embeddings, si tiene conto non solo della parola in sé ma anche dell'intera frase nella quale si trova, codificando così l'informazione del contesto.

Ci sono diverse tipologie di Transformer:

- **Encoder-decoder**: due parti separate, una legge l'input (*encoder*) e l'altra genera l'output (*decoder*). Utile per traduzione o riassunti.
- **Decoder autoregressivo**: genera una parola alla volta guardando solo le parole già prodotte (utilizzato nei modelli GPT).
- **Mixture-of-Experts (MoE)**: il modello ha molti "esperti" specializzati e per ogni input ne attiva solo alcuni, migliorando così l'efficienza [12][13][14].

1.2.4 Stadi di adattamento: dal pre-training all'uso

Il ciclo di vita di un LLM si articola tipicamente in tre fasi principali:

1. **Pre-training**: apprendimento auto-supervisionato su grandi corpus testuali per acquisire conoscenze linguistiche generali.
2. **Fine-tuning/Alignment**: adattamento a compiti specifici o preferenze umane.
3. **Prompting/Utilizzo**: impiego diretto tramite prompt.

1.3 *Pre-training*

Nella fase di *pre-training* i Large Language Models (LLM) acquisiscono una conoscenza linguistica generale e una capacità di rappresentazione del testo, prima di qualsiasi adattamento specifico.

1.3.1 Obiettivi di pre-training

Gli obiettivi di addestramento definiscono cosa il modello apprende:

- ***Full Language Modeling***: al modello viene chiesto di predire i token futuri dati i token precedenti.
- ***Prefix Language Modeling***: un insieme di alcuni token viene scelto casualmente e solo i token rimanenti devono essere previsti dal modello correttamente.
- ***Masked Language Modeling***: i token, o gli span (una sequenza di token), vengono mascherati casualmente e al modello viene chiesto di predire i token mascherati dato il contesto intorno a loro. [15]

1.3.2 Dataset di Addestramento

La qualità e la varietà dei dati di pre-training sono fondamentali. Le principali categorie di dataset includono:

- **Corpus web**: testi raccolti da siti web, ad esempio *Common Crawl*.
- **Enciclopedie e risorse strutturate**, come Wikipedia.
- **Libri e testi lunghi**, ad esempio *BooksCorpus*.
- **Forum e codice**, inclusi dataset come *The Pile*, *Github*, *StackExchange*.

Prima dell'addestramento, i dataset subiscono processi di filtraggio e deduplicazione.

1. Large Language Model - Cosa sono e come funzionano

1.3.3 Sfide del Pre-training

L'addestramento degli LLM pone sfide di tipo computazionale e metodologico:

- **Costo computazionale:** i modelli con centinaia di miliardi di parametri richiedono settimane di addestramento su supercomputer con migliaia di GPU.
- **Efficienza dei dati:** la disponibilità di dati di alta qualità è limitata.
- **Rumore e bias:** i dati web contengono rumore, contenuti tossici e bias culturali che vengono appresi dal modello.
- **Controllo della qualità:** è necessario implementare filtri e curare la selezione delle fonti.

1.4 Fine-tuning e Adattamento

Dopo il pre-training, gli LLM vengono tipicamente sottoposti a una fase di adattamento, che ne affina le capacità su compiti specifici o ne migliora l'allineamento con le intenzioni umane.

1.4.1 *Supervised Fine-tuning*

Consiste nell'addestrare ulteriormente l'LLM su un dataset etichettato per compiti specifici (e.g. traduzione, classificazione, QA).

- Viene usato per adattare un modello generalista a un dominio ristretto (e.g. medicina, diritto).
- Può essere effettuato con dataset relativamente piccoli rispetto al pre-training.
- Rischia di causare *catastrophic forgetting*, ossia perdita delle conoscenze generali.

1.4.2 *Instruction Tuning*

Gli LLM vengono ottimizzati per comprendere e seguire istruzioni in linguaggio naturale.

- Basato su dataset di tipo *prompt-response*, costruiti manualmente o tramite crowdsourcing.
- Introduce la capacità di rispondere in modo coerente a comandi generali senza ulteriori adattamenti.

[16]

1. Large Language Model - Cosa sono e come funzionano

1.4.3 *Reinforcement Learning from Human Feedback*

Il **RLHF** [17] rappresenta una tecnica fondamentale per allineare i modelli alle preferenze umane.

1. Si raccoglie un dataset di risposte generate dal modello valutate da esseri umani.
2. Si addestra un modello di reward, che impara a valutare come un essere umano.
3. L'LLM viene ulteriormente ottimizzato con tecniche di reinforcement learning.

Questo approccio è stato utilizzato, ad esempio, per ChatGPT .

1.4.4 *Domain Adaptation*

Gli LLM possono essere specializzati in domini specifici:

- ***Continued Pre-training***: ulteriore addestramento su corpus specializzati.
- ***Fine-tuning*** supervisionato su task di dominio.
- **Approcci ibridi**: combinazioni di pre-training e PEFT (*Parameter-Efficient Fine-Tuning*).

1.4.5 *Alignment Tuning*

Per garantire sicurezza, utilità e rispetto delle preferenze etiche, si introducono strategie di *alignment*:

- ***Constitutional AI***: uso di principi predefiniti come linee guida per il comportamento del modello. Questa tecnica è utilizzata da Anthropic [18].
- ***Self-critique e revisione***: i modelli generano e valutano autonomamente le proprie risposte [19][20].

1.5 Tecniche di Prompting e In-Context Learning

1.5.1 Zero-shot Prompting

Nel *zero-shot prompting* il modello riceve soltanto una descrizione del compito in linguaggio naturale, senza esempi.

- Dimostra la capacità di generalizzazione appresa durante il pre-training.
- Funziona bene per compiti semplici e largamente rappresentati nei dati di training.
- È la modalità più naturale di interazione con un LLM.

1.5.2 Few-shot Prompting e In-Context Learning

Nel *few-shot prompting*, al modello vengono forniti alcuni esempi di input-output all'interno del prompt. Consente al modello di inferire lo schema del compito e adattarsi in tempo reale. Questo comportamento è noto come *in-context learning* (ICL) [21].

1.5.3 Prompt Engineering

La formulazione del prompt influenza drasticamente la qualità delle risposte. Sono dunque emerse tecniche sistematiche di *prompt engineering*:

- *Instruction-style prompts*: forniscono istruzioni chiare e contestualizzate.
- *Template-based prompting*: utilizzo di schemi predefiniti per generare input coerenti.
- *Role prompting*: assegnazione di un "ruolo" al modello (e.g. "Sei un medico con decenni di esperienza...").

1.5.4 Prompting per il Ragionamento

Gli LLM possono essere spinti a mostrare capacità di ragionamento tramite prompt opportuni. Le tecniche principali sono:

- *Chain-of-Thought (CoT)*: incoraggia il modello a produrre spiegazioni passo-passo [22].
- *Self-Consistency*: genera più catene di ragionamento e seleziona la risposta più frequente [23].
- *Tree-of-Thought (ToT)*: esplora diversi percorsi di ragionamento, con possibilità di backtracking [24].
- *Generated Knowledge Prompting*: il modello produce prima conoscenza di supporto, poi la usa per rispondere.

1.5.5 Multi-turn Prompting

Alcuni compiti complessi richiedono interazioni iterative con il modello.

- *Single-turn prompting*: il modello riceve tutte le informazioni in un unico prompt.
- *Multi-turn prompting*: l'interazione è suddivisa in più turni, con feedback e aggiustamenti progressivi. Questo approccio è alla base degli **agenti autonomi** basati su LLM.

1.5.6 Tecniche Avanzate di Prompting

Sono state sviluppate ulteriori strategie per aumentare l'affidabilità:

- *Prompt Chaining*: suddividere un compito complesso in più sotto-compiti concatenati.

1. Large Language Model - Cosa sono e come funzionano

- ***Automatic Prompt Generation***: generazione automatica di prompt efficaci tramite ottimizzazione o meta-modelli.
- ***Retrieval-Augmented Prompting***: combinazione con sistemi di recupero documentale per fornire contesto aggiornato e ridurre le allucinazioni.

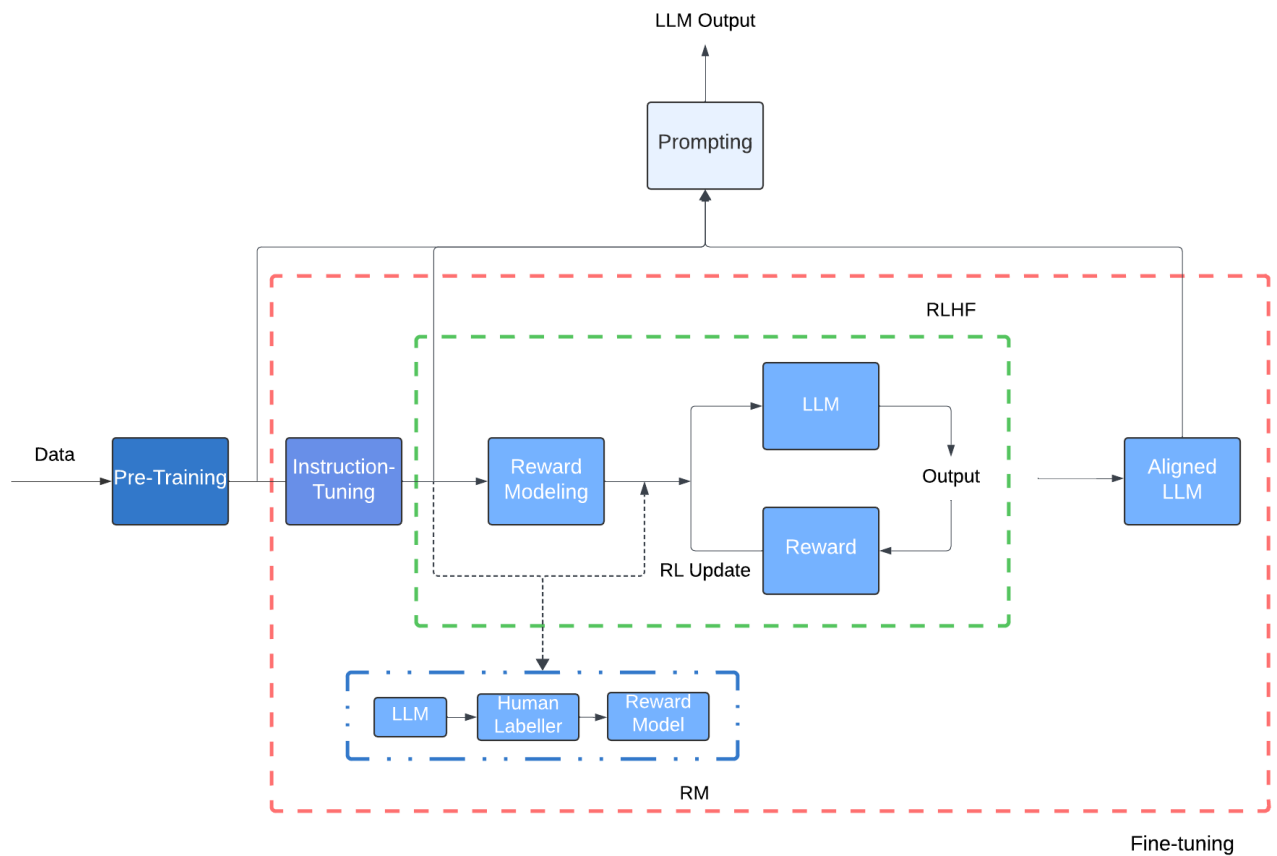


Figura 2: Un diagramma di flusso semplificato che illustra le varie fasi degli LLM, dal pre-training all'utilizzo. È possibile sollecitare gli LLM a generare risposte in diverse fasi dell'addestramento, come nel pre-training o nel fine-tuning [3].

1.6 Applicazioni

I **Large Language Models** (LLM) hanno trovato applicazione in un'ampia gamma di domini, dalla generazione di linguaggio naturale alla programmazione, dalla ricerca scientifica alla medicina.

1.6.1 Generazione e Comprensione del Linguaggio Naturale

Gli LLM eccellono in compiti tradizionali di NLP:

- **Traduzione automatica:** modelli come GPT, T5 e mBART hanno raggiunto performance vicine ai sistemi dedicati.
- **Riassunto testuale:** generazione di sintesi coerenti e concise da testi lunghi.
- **Parafrasi e riformulazione:** utile per la scrittura assistita e l'elaborazione creativa.
- **Conversazione e chatbot:** applicazioni come ChatGPT e Claude forniscono assistenza virtuale multi-dominio.

1.6.2 Programmazione e Ingegneria del Software

Gli LLM specializzati sul codice hanno mostrato grande impatto nello sviluppo software:

- **Code completion:** completamento automatico di funzioni e snippet.
- **Code generation:** traduzione da linguaggio naturale a codice eseguibile.
- **Code explanation e refactoring:** spiegazione e ottimizzazione di codice esistente.
- Strumenti come Codex, Copilot e AlphaCode hanno reso accessibile la programmazione anche a utenti non esperti.

1. Large Language Model - Cosa sono e come funzionano

1.6.3 Supporto alla Ricerca e alla Scienza

Gli LLM vengono sempre più utilizzati come strumenti di accelerazione scientifica:

- **Recupero e sintesi della letteratura scientifica.**
- **Generazione di ipotesi:** suggerimenti per esperimenti o linee di ricerca.
- **Analisi dati e interpretazione:** supporto a scienziati in fisica, biologia, chimica.
- **Scoperta di molecole e farmaci:** integrazione con modelli di chimica computazionale.

1.6.4 Medicina e Assistenza Sanitaria

Il settore medico rappresenta un campo sensibile ma promettente:

- **Supporto diagnostico:** analisi di sintomi e cartelle cliniche.
- **Assistenza alla scrittura clinica:** redazione automatica di referti.
- **Educazione medica:** generazione di materiali di studio personalizzati.
- **Chatbot sanitari:** fornire informazioni mediche preliminari ai pazienti.

Sono necessarie forti misure di sicurezza per ridurre rischi di allucinazioni e bias.

1.6.5 Educazione e Apprendimento

Gli LLM possono agire da tutor personalizzati:

- **Creazione di materiali didattici adattivi.**
- **Assistenza individuale agli studenti** con spiegazioni su misura.
- **Valutazione automatizzata** di esercizi e saggi.
- Supporto all'apprendimento delle lingue attraverso dialoghi interattivi.

1. Large Language Model - Cosa sono e come funzionano

1.6.6 Business e Produttività

Le applicazioni aziendali sono tra le più immediate:

- **Automazione del customer service** con chatbot multilingue.
- **Generazione di report e analisi** a partire da dati aziendali.
- **Supporto alle decisioni** tramite analisi predittive.
- **Content creation**: marketing, copywriting e produzione di contenuti digitali.

1.6.7 Applicazioni Multimodali

Gli LLM vengono estesi per gestire input multimodali:

- **Vision-Language Models**, come *CLIP*, *Flamingo*, GPT-4 multimodale.
- **Image captioning e visual question answering**.
- **Audio e speech**: trascrizione, traduzione e sintesi vocale.
- **Robotica**: uso di LLM per interpretare istruzioni linguistiche e guidare azioni fisiche.

1.6.8 Agenti Autonomi e Tool Use

Un campo emergente riguarda l'integrazione degli LLM in agenti autonomi:

- **Tool-augmented LLMs**: modelli che interagiscono con strumenti esterni come motori di ricerca, calcolatrici o database.
- **Agenti multi-step**: in grado di pianificare sequenze di azioni.
- Framework come *LangChain*, *AutoGPT* e *BabyAGI* mostrano le potenzialità degli LLM come agenti generici.

1.7 Sfide, Limitazioni e Aspetti Etici

Nonostante i notevoli progressi, gli LLM presentano ancora numerose sfide tecniche, limitazioni pratiche e rischi etici.

1.7.1 Allucinazioni e Aderenza ai Fatti

Gli LLM spesso generano contenuti plausibili ma falsi, un fenomeno noto come **allucinazioni**. Possono inventare dati, riferimenti bibliografici o fatti storici inesistenti. Ciò riduce l'affidabilità in contesti critici come medicina, diritto o scienza. Tecniche come *retrieval-augmented generation* (RAG) sono state proposte per mitigare questo problema.

1.7.2 Bias ed Equità

Gli LLM ereditano i bias presenti nei dati di addestramento.

- Riproducono stereotipi di genere, etnia, religione, orientamento sessuale.
- Possono amplificare discriminazioni già esistenti nella società.

Sono in corso ricerche su dataset bilanciati e metodi per eliminare i bias.

1.7.3 Sicurezza e Abusi Potenziali

Gli LLM possono essere impiegati impropriamente per generare contenuti dannosi.

- **Disinformazione:** generazione automatica di fake news e propaganda.
- **Cybersecurity:** creazione di codice malevolo o phishing su larga scala.
- **Deepfakes multimodali:** se integrati con modelli di immagini o audio.

1. Large Language Model - Cosa sono e come funzionano

1.7.4 Interpretabilità e Trasparenza

Un limite importante riguarda la natura **black-box** degli LLM.

- È difficile spiegare come vengano prese le decisioni interne al modello.
- La mancanza di interpretabilità riduce la fiducia in applicazioni sensibili.

Sono state proposte tecniche di *model probing*, analisi delle rappresentazioni interne e spiegazioni post-hoc.

1.7.5 Sostenibilità e Impatto Ambientale

Il training di LLM richiede un enorme consumo energetico.

- Addestrare un singolo modello può generare tonnellate di CO₂.
- Questo solleva preoccupazioni ambientali e di sostenibilità.

Approcci di efficientamento (quantizzazione, pruning, distillazione) mirano a ridurre tali costi.

1.7.6 Accessibilità e Disuguaglianze

Gli LLM di punta sono sviluppati principalmente da grandi aziende tecnologiche.

- L'alto costo computazionale limita la ricerca accademica indipendente.
- Si rischia una concentrazione di potere e conoscenza in poche mani.

Iniziative open-source (e.g. *LLaMA*, *Falcon*, *MPT*) cercano di democratizzare l'accesso.

1. Large Language Model - Cosa sono e come funzionano

1.7.7 Allineamento con Valori Umani

Un'altra sfida riguarda l'**alignment** con preferenze etiche e valori umani.

- Gli LLM devono essere utili, onesti e innocui.
- Tecniche come **RLHF** e *Constitutional AI* sono state sviluppate per migliorare l'allineamento.
- Rimane complesso garantire coerenza su culture e comunità diverse.

1.7.8 Aspetti Legali e Regolamentazione

Con la diffusione degli LLM emergono questioni legali:

- **Copyright:** rischio di violazioni dovute a testi generati simili a fonti protette.
- **Privacy:** possibilità che vengano riprodotte informazioni sensibili dai dati di training.
- **Regolamentazione:** discussioni in corso a livello internazionale su AI Act e framework etici.

1.8 Prospettive Future e Direzioni di Ricerca

Gli LLM rappresentano uno dei campi più dinamici dell'intelligenza artificiale. Dopo i progressi ottenuti negli ultimi anni, la comunità scientifica si interroga su come evolveranno i modelli futuri e quali direzioni di ricerca saranno più rilevanti. Questo capitolo discute le prospettive principali.

1.8.1 Efficienza e Sostenibilità

Una priorità sarà lo sviluppo di modelli più efficienti dal punto di vista computazionale ed energetico.

- Tecniche di compressione (quantizzazione, pruning, distillazione) sempre più sofisticate.
- Architetture ibride che combinano Transformer con nuovi meccanismi di attenzione.
- Algoritmi di addestramento ottimizzati per ridurre costi e tempi.

1.8.2 Modelli Multimodali

Gli LLM tenderanno a integrarsi con altre modalità sensoriali:

- **Vision-Language Models:** unione di testo e immagini.
- **Speech e audio:** modelli in grado di comprendere e generare linguaggio parlato.
- **Video e robotica:** comprensione di sequenze visive e comandi linguistici per il controllo di agenti fisici.

L'obiettivo è costruire **modelli universali** in grado di gestire input multimodali.

1. Large Language Model - Cosa sono e come funzionano

1.8.3 Memory-Augmented Large Language Models

Attualmente gli LLM hanno memoria limitata alla finestra contestuale.

- Ricerca su architetture con memoria esterna persistente.
- Meccanismi per richiamare informazioni apprese in sessioni precedenti.
- Possibilità di modelli che accumulano conoscenza nel tempo, riducendo la necessità di addestrarli di nuovo in modo completo.

1.8.4 Autonomia e Agenti Intelligenti

Un filone emergente è l'integrazione degli LLM in agenti autonomi.

- Capacità di pianificazione multi-step tramite prompting avanzato.
- Integrazione con tool esterni: database, motori di ricerca, calcolatrici, API.
- Applicazioni in domini complessi come finanza, ricerca scientifica, sviluppo software.

1.8.5 Interpretabilità e Controllo

La comunità scientifica riconosce la necessità di modelli più trasparenti:

- Strumenti di **explainable AI** per comprendere le decisioni interne.
- Analisi delle rappresentazioni neurali per capire l'emergere delle abilità.
- Tecniche di controllo *fine-grained* per modulare il comportamento del modello.

1.8.6 Sicurezza e Robustezza

Un'altra direzione riguarda la costruzione di LLM affidabili e sicuri:

- Difese contro attacchi avversari e *prompt injection*.

1. Large Language Model - Cosa sono e come funzionano

- Riduzione delle allucinazioni.
- Maggiore robustezza a input rumorosi o manipolati.

1.8.7 Allineamento e Governance

Il tema dell'allineamento con valori umani resterà centrale.

- Evoluzione di tecniche come RLHF, Constitutional AI e feedback multimodale.
- Creazione di framework etici condivisi per la progettazione di modelli sicuri.
- Discussione su regolamentazione internazionale e open science.

1.8.8 Verso l'Intelligenza Artificiale Generale

Infine, gli LLM sono considerati possibili passi verso l'AGI, *Artificial General Intelligence*.

- Studi sull'emergere di capacità generali di ragionamento e apprendimento.
- Integrazione con sistemi simbolici e neuro-simbolici.
- Possibilità di architetture future che superino i limiti dei Transformer attuali.

2

Model Context Protocol

2.1 Funzionamento

Il Model Context Protocol (MCP) è un protocollo open-source che standardizza il modo in cui le applicazioni forniscono contesto agli LLM. Per analogia, si può pensare a MCP come a una porta USB-C per le applicazioni IA. Proprio come USB-C fornisce un modo standardizzato per collegare i dispositivi a varie periferiche e accessori, MCP fornisce un modo standardizzato per collegare le IA a diverse fonti di dati e strumenti. Il protocollo consente di creare agenti e flussi di lavoro complessi basati sugli LLM e connette i modelli al resto del mondo informatico. [25]

Il protocollo MCP include i seguenti progetti:

- **Specifiche MCP**, che delineano i requisiti di implementazione per client e server.
- **SDK MCP**: *Software Development Kit* (SDK) per diversi linguaggi di programmazione che implementano MCP.
- **Strumenti di sviluppo MCP**.
- **Implementazioni di server MCP di riferimento**.

2. Model Context Protocol

MCP si concentra esclusivamente sul protocollo per lo scambio di contesto, senza stabilire come le applicazioni IA utilizzino gli LLM o gestiscano il contesto fornito. [26]

2.1.1 Partecipanti

Il protocollo segue un'architettura client-server in cui un Host stabilisce connessioni a uno o più Server creando un Client dedicato per ciascuno di essi. I principali partecipanti all'architettura MCP sono:

- **Host MCP:** l'applicazione IA che coordina e gestisce uno o più Client.
- **Client MCP:** un componente che mantiene una connessione a un Server e ottiene da esso il contesto che l'Host può utilizzare.
- **Server MCP:** un programma che fornisce contesto ai Client; possono essere eseguiti sia in locale che in remoto. [26]

2.1.2 Livelli

MCP è costituito da due livelli:

- **Livello dati:** definisce il protocollo, basato su JSON-RPC, per la comunicazione client-server, inclusa la gestione del ciclo di vita e le primitive principali, come strumenti, risorse, prompt e notifiche.
- **Livello trasporto:** definisce i meccanismi e i canali di comunicazione che consentono lo scambio di dati tra client e server, inclusi l'instaurazione di connessioni specifiche per il trasporto, il framing dei messaggi e l'autorizzazione.

Concettualmente, il livello dati è il livello interno, mentre il livello trasporto è il livello esterno. [26]

2. Model Context Protocol

Livello dati

Il livello dati implementa un protocollo di scambio basato su JSON-RPC 2.0 che definisce la struttura e la semantica dei messaggi. Questo livello include:

- Gestione del ciclo di vita: gestisce l'inizializzazione della connessione, la negoziazione delle capacità e la terminazione della connessione tra client e server.
- Funzionalità del server: consente ai server di fornire funzionalità di base, inclusi strumenti per azioni IA, risorse per dati di contesto e richieste per schemi di interazione da e verso il client.
- Funzionalità del client: consente ai server di chiedere al client di campionare dall'LLM, ottenere input dall'utente e registrare messaggi al client.
- Funzionalità di utilità: supporta funzionalità aggiuntive come notifiche per aggiornamenti in tempo reale e monitoraggio dei progressi per operazioni di lunga durata. [26]

Livello di trasporto

Il livello di trasporto gestisce i canali di comunicazione e l'autenticazione tra client e server. Si occupa della creazione della connessione, il framing dei messaggi e la comunicazione sicura tra i partecipanti. MCP supporta due meccanismi di trasporto:

- *Stdio Transport*: utilizza flussi di input/output standard per la comunicazione diretta tra processi locali sulla stessa macchina, garantendo prestazioni ottimali senza sovraccarico di rete.
- *Streamable HTTP transport*: utilizza metodi HTTP POST per i messaggi client-server, inclusi eventi opzionali inviati dal server per le funzionalità di streaming. Questo trasporto consente la comunicazione con il server remoto e supporta metodi di autenticazione HTTP standard, inclusi *bearer token*, chiavi API e intestazioni personalizzate.

2. Model Context Protocol

MCP consiglia di utilizzare OAuth per ottenere i token di autenticazione. Il livello di trasporto astrae i dettagli di comunicazione dal livello di protocollo, consentendo lo stesso formato di messaggio del protocollo JSON-RPC 2.0 su tutti i meccanismi di trasporto. [26]

2.1.3 Protocollo del livello dati

Elemento fondamentale del Model Context Protocol è la definizione dello schema e della semantica tra Client e Server. Il livello dati è la parte del protocollo che si occupa di definire le modalità con cui gli sviluppatori possono condividere il contesto dai Server ai Client. MCP utilizza JSON-RPC 2.0 come protocollo *Remote Procedure Call* (RPC). Client e Server si inviano richieste e rispondono di conseguenza. Le notifiche possono essere utilizzate quando non è richiesta alcuna risposta. [26]

Primitive

Le primitive MCP sono il concetto più importante perché definiscono ciò che Client e Server possono offrirsi reciprocamente. Queste primitive specificano i tipi di informazioni contestuali che possono essere condivise con le applicazioni IA e la gamma di azioni che possono essere eseguite. MCP definisce tre primitive principali che i server possono esporre:

- *Tools*: funzioni eseguibili che le applicazioni IA possono invocare per eseguire azioni (e.g. operazioni su file, chiamate API, query di database).
- *Risorse*: fonti di dati che forniscono informazioni contestuali alle applicazioni IA (e.g. contenuto di file, record di database, risposte API).
- *Prompt*: schemi riutilizzabili che aiutano a strutturare le interazioni con i modelli linguistici (e.g. prompt di sistema, prompt *few-shot*).

Ogni tipo di primitiva ha metodi associati per la scoperta (`*/list`), il recupero (`*/get`) e, nel caso dei tools, l'esecuzione (`tools/call`). Ad esempio, un Client può

2. Model Context Protocol

prima elencare tutti i tools disponibili (`tools/list`) e poi eseguire quelli che ritiene necessari. Questa progettazione consente di creare elenchi dinamici.

Come esempio concreto, si consideri un Server che fornisce contesto su un database. Può esporre strumenti per interrogare il database, una risorsa che contiene lo schema del database e un prompt che include esempi di interazione con gli strumenti.

MCP definisce anche le primitive che i Client possono esporre. Queste primitive consentono agli autori del Server di creare interazioni più ricche.

- **Campionamento:** consente ai Server di richiedere il completamento di un testo da parte di un modello linguistico dall'applicazione IA del Client. Questa funzionalità è utile quando gli autori del Server desiderano accedere a un modello linguistico senza però includere un SDK associato. Possono utilizzare il metodo `sampling/complete`.
- **Elicitazione:** consente ai Server di richiedere informazioni aggiuntive agli utenti. Questa funzionalità è utile quando gli autori del Server desiderano ottenere maggiori informazioni dall'utente o chiedere la conferma di un'azione. Possono utilizzare il metodo `elicitation/request`.
- **Logging:** consente ai Server di inviare messaggi di log ai Client a scopo di debug e monitoraggio. [26]

Notifiche

Il protocollo supporta notifiche in tempo reale per abilitare aggiornamenti dinamici tra Server e Client. Ad esempio, quando gli strumenti disponibili su un Server cambiano, cosa che accade quando vengono rese disponibili nuove funzionalità o quando vengono modificati strumenti esistenti, esso può inviare notifiche di aggiornamento ai Client connessi per informarli. Le notifiche vengono inviate come messaggi di notifica JSON-RPC 2.0 (senza attendere una risposta). [26]

3

Ticket Management System

Il *Ticket Management System* è un'applicazione che ha lo scopo di aiutare gli utenti nella creazione e gestione di ticket, creati per segnalare la presenza di problemi individuati nell'uso di un ipotetico sistema software, e nel loro inoltro a degli sviluppatori che possano analizzare e poi risolvere tali problemi. La funzionalità principale dell'applicazione risiede nella possibilità, per l'utente, di interfacciarsi con un chatbot al fine di usufruire dei servizi offerti.

3.0.1 Scelta del Large Language Model

In una prima bozza, l'LLM scelto per comunicare con l'utente era Claude Sonnet. Il progetto era stato configurato per rendere visibile i suoi tools all'applicazione Claude Desktop, cosicché potesse essere utilizzato direttamente all'interno del software in questione. La bozza ha avuto successo, ma essendo la solution pensata per essere integrata nell'ambiente cloud di Microsoft Azure, era indispensabile un differente LLM, il quale non fosse limitato a un programma esterno.

Per questo motivo è stato scelto un chatbot interno all'ambiente Azure, ossia una risorsa Azure OpenAI [27], che mette a disposizione una serie di modelli proprietari di OpenAI, tra cui GPT-4.1, infine scelto per il progetto definitivo.

3. Ticket Management System

3.1 Struttura del progetto

Il *Ticket Management System* è stato progettato come una *solution* in C# suddivisa, secondo un approccio modulare, in più progetti indipendenti ma interconnessi. Tale organizzazione riflette i principi delle moderne architetture software a livelli, secondo il principio *Separation of Concerns* [28], favorendo la manutenibilità e la possibilità di estendere il sistema senza introdurre dipendenze circolari o accoppiamenti eccessivamente rigidi.

La composizione della *solution* prevede cinque progetti principali, ciascuno con un ruolo ben definito: *TM.Shared*, *TM.Data*, *TM.CQRS*, *TM.Function* e *TM.Client*.

3.1.1 TM.Shared

Il progetto *TM.Shared* definisce tutte le classi degli oggetti che vengono creati internamente nel codice a seguito dell'interazione con l'utente. Tali classi sono:

- **Ticket**: elemento centrale del sistema; rappresenta i ticket creati dagli utenti, con i relativi **Comment**, le **Task** a esso legate, il **TicketStatus** e la **TicketPriority**.
- **Category**: ogni **Ticket** appartiene a una categoria.
- **Comment**: i commenti associati a un determinato **Ticket**.
- **Task**: i task sono i compiti affidati agli sviluppatori, generati dal sistema a seguito della creazione di un **Ticket**.
- **User**: gli utenti che usano il sistema; sono intesi anche gli sviluppatori che devono risolvere le **Task**.

Oltre alle classi, il progetto definisce dei DTO (*Data Transfer Objects*) [29]. I DTO sono classi compatte che assomigliano alle classi originali ma che contengono meno informazioni, limitandosi a quelle ritenute necessarie. L'obiettivo è migliorare le performance legate al trasferimento dati tra i diversi sottosistemi, oltre a nascondere alcune logiche interne delle varie classi.

3. Ticket Management System

3.1.2 TM.Data

Il progetto *TM.Data* ha il compito di gestire la persistenza delle informazioni, fornendo l'accesso al database e archiviando gli oggetti creati. Il database, di tipo relazionale, è archiviato all'interno dell'ambiente cloud Azure ed il linguaggio utilizzato per interrogarlo è SQL.

3.1.3 TM.CQRS

Il progetto *TM.CQRS* introduce l'implementazione del pattern *Command Query Responsibility Segregation* [30]. Il CQRS è un pattern architetturale che ha lo scopo di isolare le operazioni di lettura (*Queries*) e di scrittura (*Commands*) del database in un progetto separato, che possono poi essere richiamate da ognuno degli altri progetti per effettuare le interrogazioni.

3.1.4 TM.Function

Il cuore della logica applicativa è rappresentato da *TM.Function*. Essa è un'*Azure Function*, ovvero una soluzione serverless che consente agli sviluppatori di scrivere ed eseguire codice sul cloud in maniera più efficace e compatta, senza dover ricorrere alla creazione di un progetto [31]. Prende il ruolo dell'MCP Server, l'applicazione IA che gestisce la logica principale del Model Context Protocol. Verrà discusso più approfonditamente in seguito.

3.1.5 TM.Client

Infine, *TM.Client* costituisce l'interfaccia utente, nonché l'MCP Client. Esso contiene la sola classe `Program.cs`, occupandosi di avviare l'intera solution e integrare i vari altri progetti, oltre che inizializzare la comunicazione con il chatbot tramite terminale.

3. Ticket Management System

3.2 Il Model Context Protocol in pratica

Nucleo centrale del progetto è la classe `TicketManagementTools` del progetto `TM.Function`. Il suo obiettivo è quello di definire una serie di *tools* che l'LLM invoca per eseguire le operazioni richieste. Vediamo un esempio:

```
[Function("AggiungiTicket")]
public async Task<TicketDtoCreate?> AggiungiTicketAsync(
    [McpToolTrigger("AggiungiTicket", "Aggiunge un ticket")] ToolInvocationContext context,
    [McpToolProperty("Ticket", "string", DtoJsonSchemas.TicketDtoCreateSchema)] string ticketDto)
{
    [...]
}
```

Questo metodo permette di creare un oggetto `Ticket` (più precisamente, crea un'istanza della classe DTO `TicketDtoCreate`). Caratteristica del Model Context Protocol sono le decorazioni dei metodi fornite dal protocollo: `McpToolProperty` e `McpToolTrigger`. Esse esplicitano i propri dettagli operativi in linguaggio naturale, invece che in un linguaggio di programmazione, permettendo all'LLM di interpretare queste istruzioni e capire quali dei tool a sua disposizione utilizzare.

Si osservi l'utilizzo di JSON all'interno del metodo: per comunicare all'LLM la struttura degli oggetti da creare, vengono forniti in input gli schemi JSON degli oggetti in questione. Nel frammento mostrato, lo schema è fornito tramite `DtoJsonSchemas.TicketDtoCreateSchema` nell'attributo `McpToolProperty`.

Il parametro `ToolInvocationContext`, nell'esempio, fornisce il contesto dell'invocazione del tool: rappresenta il punto di integrazione tra la chiamata esterna (o generata dall'LLM) e l'implementazione del metodo. La presenza di tale contesto, e delle specifiche in linguaggio naturale nelle decorazioni, facilita la correlazione tra le istruzioni fornite al modello e le operazioni concrete eseguite dall'applicazione.

Questi elementi insieme mostrano come il Model Context Protocol consenta di esporre funzioni applicative al modello in modo strutturato. Le descrizioni in linguaggio naturale rendono comprensibili i comportamenti dei tool all'LLM, mentre

3. Ticket Management System

gli schemi JSON garantiscono che i dati prodotti rispettino le definizioni interne alla solution.

3. Ticket Management System

3.3 Verifica pratica

Per verificare il corretto funzionamento del sistema software si può procedere come segue:

1. avviare una conversazione con il chatbot, nella quale viene chiesto di creare un ticket;
2. prestare attenzione al fatto che vengano rispettati i parametri degli oggetti così come sono definiti all'interno delle classi del progetto;
3. verificare che tali oggetti vengano effettivamente creati all'interno del database nell'ambiente cloud Azure di cui prima.

Segue una trascrizione di un'interazione avuta con il chatbot volta a mettere in atto questi passaggi.

Nota: ciò che segue la scritta **Prompt:** è ciò che è stato scritto da un utente che usa il sistema, mentre il resto è la risposta generata dall'LLM.

```
Prompt: vorrei creare un ticket
Per creare un ticket ho bisogno di alcune informazioni:

1. Titolo del ticket
2. Descrizione del problema o richiesta
3. Priorità (Low, Medium, High)
4. Il tuo ID utente (o i tuoi dati se non sei ancora registrato)
5. Categoria del ticket (selezionabile da elenco categorie
disponibili)

Puoi fornirmi queste informazioni? Se non conosci la categoria o
il tuo ID utente, posso aiutarti a recuperarli.
Prompt: il titolo è "Problema nell'autenticazione"
Ottimo, hai scelto il titolo "Problema nell'autenticazione".

Ora ho bisogno delle seguenti informazioni per completare la
```

3. Ticket Management System

creazione del ticket:

1. Descrizione del problema
2. Priorità (Low, Medium, High)
3. Il tuo ID utente (oppure il tuo username/email/nome e cognome per identificarti)
4. Categoria del ticket (se non la conosci, posso mostrarti la lista delle categorie disponibili)

Per favore, forniscimi questi dettagli.

Prompt: la descrizione è "Ho problemi nell'autenticazione. Non riesco ad inserire username e password nel form."

Perfetto, ho annotato la descrizione:

"Ho problemi nell'autenticazione. Non riesco ad inserire username e password nel form."

Mi servono ancora alcune informazioni:

1. Priorità del ticket (Low, Medium, High)
2. Il tuo ID utente (oppure username/email/nome e cognome, se non conosci l'ID)
3. Categoria del ticket (se non la conosci posso elencare le categorie disponibili)

Puoi fornirmi questi dettagli? Vuoi che ti mostri le categorie disponibili?

Prompt: priorità alta

Le categorie disponibili sono:

1. Hardware: Categoria relativa a hardware e dispositivi fisici
2. Software Development: Category for software development related tasks
3. Networking: Categoria relativa alle reti di computer
4. Software Testing: Categoria per attività di testing software

3. Ticket Management System

Quale categoria vuoi assegnare al ticket? Inoltre, puoi fornirmi il tuo ID utente o, se non lo conosci, il tuo username/email/nome e cognome per identificarti?

Prompt: vorrei creare una nuova categoria

Per creare una nuova categoria, ho bisogno di:

1. Nome della categoria
2. Descrizione della categoria (opzionale, ma consigliata)

Per favore, forniscimi questi dettagli.

Prompt: "Problemi di autenticazione - Problemi legati all'autenticazione nel sistema"

La nuova categoria "Problemi di autenticazione" è stata creata con successo.

Ora, per completare il ticket, puoi fornirmi il tuo ID utente oppure il tuo username/email/nome e cognome? In questo modo potrò associare il ticket al tuo profilo.

Prompt: l'username è adelprete

Il ticket "Problema nell'autenticazione" con priorità alta è stato creato nella categoria "Problemi di autenticazione" ed è stato associato al tuo profilo (username: adelprete).

Se hai bisogno di aggiungere commenti, task, o apportare modifiche al ticket, fammelo sapere!

Come si evince dalla conversazione, l'LLM ha correttamente compreso quali sono i parametri dell'oggetto `Ticket` e, ogniqualvolta che l'utente specifica uno di questi parametri, il chatbot lo memorizza e passa a quello successivo, finché non ha ottenute tutte le informazioni necessarie per creare l'oggetto finale.

Ha per giunta reperito le categorie già esistenti all'interno del database e, alla richiesta dell'utente di crearne una nuova, l'LLM temporaneamente sospende la

3. Ticket Management System

creazione del ticket per soddisfare tale richiesta, per poi ritornare all'operazione iniziale. Questo dimostra la capacità di poter passare da un compito all'altro in modo fluido.

3. Ticket Management System

Come prova definitiva, seguono due screenshot del database sulla piattaforma Azure per osservare il contenuto della tabella *dbo.Tickets* prima e dopo la conversazione di cui sopra.

Home page > sqldb-tm-dev (sql-tm-dev/sqldb-tm-dev)

sqldb-tm-dev (sql-tm-dev/sqldb-tm-dev) | Editor di query (anteprima) ★ ...

Database SQL

» Accesso + Nuova query ↕ Apri query Commenti e suggerimenti Attività iniziali

»>>

dbo.Tickets ×

» Crea nuova riga Salva Aggiorna Rimuovi Elimina la riga

Esegui una ricerca per filtrare gli elementi...

Id	Title	Description	CreatedAt	UpdatedAt	Status	Priority	CreatedByUserId	CategoryId
----	-------	-------------	-----------	-----------	--------	----------	-----------------	------------

Figura 3: Prima della conversazione. Non ci sono ancora ticket.

Home page > sqldb-tm-dev (sql-tm-dev/sqldb-tm-dev)

sqldb-tm-dev (sql-tm-dev/sqldb-tm-dev) | Editor di query (anteprima) ★ ...

Database SQL

» Accesso + Nuova query ↕ Apri query Commenti e suggerimenti Attività iniziali

»>>

dbo.Tickets ×

» Crea nuova riga Salva Aggiorna Rimuovi Elimina la riga

Esegui una ricerca per filtrare gli elementi...

Id	Title	Description	CreatedAt	UpdatedAt	Status	Priority	CreatedByUserId	CategoryId
3	Problema nell'autenticazione	Ho problemi nell'autenticazione. Non riesco ad inserire username e password nel form.	2025-09-16...	2025-09-16...	0	2	3	7

Figura 4: Dopo la conversazione. Il ticket è stato aggiunto.
Nota: le due date e lo status sono stati generati automaticamente.

3. Ticket Management System

3.4 Limitazioni

Le limitazioni del sistema sono sostanzialmente equiparabili a quelle degli LLM discusse in precedenza. Di particolare rilevanza, nel nostro caso, è la tendenza ad allucinare informazioni. Data la necessità di creare oggetti che siano conformi ai parametri stabiliti dalle classi del progetto, un errore in questo aspetto comprometterebbe il corretto funzionamento dell'intero sistema.

Inoltre, il contesto a cui un LLM può far riferimento, la cosiddetta *context window*, è circoscritto. Ciò pone un limite a quante informazioni possono essere gestite dal chatbot all'interno di una stessa conversazione prima che le sue capacità inizino a degradare [32]. Questo può diventare particolarmente rilevante nel momento in cui il Ticket Management System verrà integrato in un sistema software più vasto, come discusso nella conclusione di questa tesi, perché ciò richiederà la gestione di una potenzialmente grande quantità di informazioni. Potrebbe essere necessario sviluppare una procedura più sistematica e ottimizzata per acquisire delle informazioni utili in un particolare momento oppure imporre una dimensione massima per le chat.

Conclusioni e sviluppi futuri

Il *Ticket Management System* è pensato per essere posizionato all'interno di un contesto più grande, ossia all'interno di un sistema software per il quale un utente può avere necessità di un supporto tecnico. Il sistema che verrà preso in esame è 3FS[©].

3FS[©] - Panoramica del funzionamento

3FS[©] (*Fleet and Financial Flexible System*) è una piattaforma avanzata per la gestione delle flotte aziendali, che automatizza processi operativi, commerciali, contabili e normativi. Ottimizza il ciclo di vita dei veicoli gestendo accordi, rotazione, acquisti, configurazione della flotta e distribuzione, con monitoraggio logistico e riduzione dei rischi di non conformità.

Il sistema automatizza adempimenti come registrazione targhe, tasse, assicurazioni e rende i veicoli rapidamente disponibili al noleggio. Gestisce manutenzione, rinnovi, tasse, sinistri e interagisce con le autorità per la gestione delle multe. Supporta la dismissione dei veicoli con processi automatizzati e integrazione contabile.

Sul piano finanziario, consente la registrazione e riconciliazione in tempo reale di fatture, gestione dei flussi di cassa e reportistica, integrandosi con sistemi contabili esterni per maggiore controllo e trasparenza. [33]

Ticket Management System applicato a 3FS[©]

Il Ticket Management System è per ora solo un proof-of-concept ma, in futuro, si potrà integrare all'interno di 3FS[©] per fornire supporto tecnico agli utenti. I ticket saranno dunque inerenti a problemi legati alla piattaforma in questione, il che presuppone che l'LLM abbia una conoscenza dettagliata del suo funzionamento, anche per poter rispondere in modo autonomo a problematiche semplici.

Vantaggi attesi

L'integrazione del Ticket Management System con 3FS[©] mira a ottenere diversi benefici:

- riduzione del tempo medio di risoluzione dei problemi;
- diminuzione del carico di lavoro che grava sugli operatori umani;
- miglioramento della qualità del servizio e della soddisfazione dell'utente;

Conclusione

Il Model Context Protocol offre un nuovo modo per potenziare le capacità degli LLM, già di per sé strumenti estremamente capaci e potenti. Questa tesi ha illustrato una possibile applicazione di questa tecnologia. L'uso di questo protocollo può permettere nuove forme di interazione tra modelli linguistici e sistemi aziendali complessi, aprendo la strada a un'integrazione sempre più forte tra intelligenza artificiale e processi produttivi.

Bibliografia

- [1] UBS. Latest House View Daily, 2023. URL: <https://www.ubs.com/global/en/wealthmanagement/insights/chief-investment-office/house-view/daily/2023/latest-25052023.html>. Citato a pagina 6.
- [2] Model Context Protocol. Getting Started - Introduction, 2024. URL: <https://modelcontextprotocol.io/docs/getting-started/intro>. Citato a pagina 7.
- [3] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. arXiv preprint arXiv:2307.06435, 2023. URL: <https://arxiv.org/abs/2307.06435>, arXiv:2307.06435. Citato alle pagine 8 and 18.
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL: <https://arxiv.org/abs/1810.04805>, arXiv:arXiv:1810.04805. Citato a pagina 9.
- [5] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237. Association for Computational Lin-

Bibliografia

- guistics, 2018. URL: <https://aclanthology.org/N18-1202>. Citato a pagina 9.
- [6] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019. URL: <https://arxiv.org/abs/1910.13461>, arXiv:arXiv:1910.13461. Citato a pagina 9.
- [7] Andrey Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. Transformers: "the end of history" for natural language processing? In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III*, volume 12977 of *Lecture Notes in Computer Science*, pages 677–693. Springer, 2021. doi:10.1007/978-3-030-86523-8_41. Citato a pagina 9.
- [8] J. J. Webster and C. Kit. Tokenization as the initial phase in NLP. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING), Volume 4*, 1992. Citato a pagina 10.
- [9] Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, 2018. URL: <https://aclanthology.org/P18-1007>. Citato a pagina 10.
- [10] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016. URL: <https://aclanthology.org/P16-1162>. Citato a pagina 10.

Bibliografia

- [11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. Archived (PDF) from the original on 2024-02-21. Retrieved 2024-01-21. URL: <https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>. Citato a pagina 11.
- [12] William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research*, 23(1):5232–5270, 2022. Citato a pagina 11.
- [13] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, et al. GLAM: Efficient Scaling of Language Models with Mixture-of-Experts. In *Proceedings of the International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022. Citato a pagina 11.
- [14] X. Ren, P. Zhou, X. Meng, X. Huang, Y. Wang, W. Wang, P. Li, X. Zhang, A. Podolskiy, G. Arshinov, et al. Pangu-P: Towards Trillion Parameter Language Model with Sparse Heterogeneous Computing, 2023. URL: <https://arxiv.org/abs/2303.10845>, arXiv:arXiv:2303.10845. Citato a pagina 11.
- [15] Tao Wang, Adam Roberts, David Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, J  r  my Launay, and Colin Raffel. What Language Model Architecture and Pretraining Objective Works Best for Zero-Shot Generalization? In *Proceedings of the International Conference on Machine Learning*, pages 22964–22984. PMLR, 2022. URL: <https://proceedings.mlr.press/v162/wang22a.html>. Citato a pagina 12.
- [16] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Saurabh Brahma, et al. Scaling instruction-finetuned language models, 2022. URL: <https://arxiv.org/abs/2210.12178>.

Bibliografia

- `//arxiv.org/abs/2210.11416`, `arXiv:arXiv:2210.11416`. Citato a pagina 14.
- [17] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint*, 2019. URL: <https://arxiv.org/abs/1909.08593>, `arXiv:1909.08593`. Citato a pagina 15.
- [18] Anthropic. Claude’s constitution, 2023. URL: <https://www.anthropic.com/news/claudes-constitution>. Citato a pagina 15.
- [19] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Sharan Batra, Puneet Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL: <https://arxiv.org/abs/2307.09288>, `arXiv:arXiv:2307.09288`. Citato a pagina 15.
- [20] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 27730–27744, 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html. Citato a pagina 15.
- [21] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui. A survey for in-context learning, 2023. URL: <https://arxiv.org/abs/2301.00234>, `arXiv:2301.00234`. Citato a pagina 16.
- [22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H. Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022. Citato a pagina 17.

Bibliografia

- [23] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2022. URL: <https://arxiv.org/abs/2203.11171>, arXiv:2203.11171. Citato a pagina 17.
- [24] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL: <https://arxiv.org/abs/2305.10601>, arXiv:2305.10601. Citato a pagina 17.
- [25] Model Context Protocol. Getting Started - Introduction, 2024. URL: <https://modelcontextprotocol.io/docs/getting-started/intro>. Citato a pagina 28.
- [26] Model Context Protocol. Learn - Architecture, 2024. URL: <https://modelcontextprotocol.io/docs/learn/architecture>. Citato alle pagine 29, 30, 31, and 32.
- [27] Microsoft Docs. What is azure ai foundry for openai?, 2024. URL: <https://learn.microsoft.com/en-us/azure/ai-foundry/openai/overview>. Citato a pagina 33.
- [28] MSDN Magazine Brownfield. Separation of concerns: A brownfield development series, 2009. URL: <https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/brownfield/separation-of-concerns-a-brownfield-development-series>. Citato a pagina 34.
- [29] Microsoft Docs. Using Web API with Entity Framework (Part 5). Microsoft Docs. URL: <https://learn.microsoft.com/it-it/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5>. Citato a pagina 34.

Bibliografia

- [30] Microsoft Docs. CQRS pattern. Microsoft Docs. URL: <https://learn.microsoft.com/it-it/azure/architecture/patterns/cqrs>. Citato a pagina 35.
- [31] Microsoft Docs. Azure Functions overview. URL: <https://learn.microsoft.com/it-it/azure/azure-functions/functions-overview?pivots=programming-language-csharp>. Citato a pagina 35.
- [32] IBM. What is a context window?, 2024. URL: <https://www.ibm.com/think/topics/context-window>. Citato a pagina 43.
- [33] ProgeSoftware. 3FS - ProgeSoftware, 2021. URL: <https://3fs.progesoftware.it/>. Citato a pagina 44.