Dipartimento di Ingegneria Civile, Informatica
e delle Tecnologie Aeronautiche

Corso di Laurea in Ingegneria Informatica

Aumentare le capacità di un chatbot usando Model Context Protocol (MCP)

Anno Accademico 2024/2025

Laureando

Del Prete Andrea

Matricola 589453

Relatore

Prof. Merialdo Paolo

Tutor

Dott. Di Nardo Giorgio

Indice

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 di ChatGPT, che infranse il record di applicazione con la più rapida crescita nella storia di Internet, ben 100 milioni di utenti in soli 2 mesi [1]. Gli ultimi anni in particolare sono stati caratterizzati dall'intelligenza artificiale generativa, capace cioè di generare testi, audio, immagini o video a partire da un semplice "prompt" dell'utente, come ad esempio i modelli GPT alla base di ChatGPT stesso.

Di particolare interesse sono le IA capaci di generare testo, i cosiddetti Large Language Model (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 al pari di 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 anni è perciò nato il desiderio di rendere ancora più sofisticate le abilità di questi modelli linguistici, finora relegate al "semplice" rispondere ai messaggi degli utenti. Ed è per questo che Anthropic, una delle maggiori aziende nel settore della ricerca sugli LLM e creatrice dei molto diffusi modelli Claude, ha ideato un meccanismo che facilita questo obiettivo: il Model Context Protocol (MCP). Questo

protocollo open-source definisce una procedura standard che consente alle applicazioni di fornire contesto ai modelli linguistici, permettendo loro di accedere ad ed interagire con una serie di strumenti esterni 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].

Obiettivo di questa tesi è illustrare un progetto volto a dimostrare nella pratica le funzionalità del Model Context Protocol. Il progetto in questione è un sistema di ticket management, che consente agli utenti di interfacciarsi con un chatbot che rende automatica la creazione di ticket delineanti i problemi riscontrarti e l'inoltro degli stessi a degli sviluppatori, che possono visionarli ed avviare delle procedure per risolverli.

1

Large Language Model - Cosa sono e come funzionano

1.1 Breve storia

All'inizio degli anni '90, i modelli statistici di IBM hanno aperto la strada alle tecniche di allineamento delle parole per la traduzione automatica. Durante gli anni 2000, con l'aumento dell'accesso diffuso a Internet, i ricercatori hanno iniziato a compilare enormi set di dati testuali dal web ("web as corpus" [3]) per addestrare questi modelli linguistici statistici [4][5], chiamati modelli n-gram [6].
Andando oltre i modelli n-gram, nel 2000 i ricercatori hanno iniziato a utilizzare le reti neurali per addestrare i modelli linguistici [7]. In seguito al successo delle reti neurali profonde ("Deep Neural Networks") nella classificazione delle immagini intorno al 2012 [8], architetture simili sono state adattate per compiti linguistici. Questo cambiamento è stato segnato dallo sviluppo di "word embeddings" (ad esempio, Word2Vec di Mikolov nel 2013) e modelli "seq2seq". Nel 2016, Google ha convertito il suo servizio di traduzione alla traduzione automatica neurale ("Neural Machine Translation"), sostituendo i modelli statistici basati su frasi con reti neurali profonde ricorrenti ("Deep Recurrent Neural Networks").

Alla conferenza NeurIPS del 2017, i ricercatori di Google hanno introdotto l'architettura del trasformatore in un articolo, divenuto fondamentale nel settore, chiamato "Attention Is All You Need" [9]. L'obiettivo di questo articolo era quello di migliorare la tecnologia seq2seq del 2014 basandosi su un meccanismo chiamato attenzione, sviluppato nel 2014 [10]. L'anno successivo, nel 2018, è stato introdotto BERT, che è rapidamente diventato "onnipresente" [11]. Sebbene il trasformatore originale abbia sia blocchi encoder che decoder, BERT è un modello solo encoder. L'uso accademico e di ricerca di BERT ha iniziato a diminuire nel 2023, a seguito di rapidi miglioramenti nelle capacità dei modelli solo decoder (come GPT) di risolvere compiti tramite prompt [12].
Sebbene GPT-1 ("Generative Pre-Trained Transformer"), modello solo decoder, sia stato introdotto nel 2018, è stato GPT-2 nel 2019 ad attirare l'attenzione generale, poichè OpenAI ha affermato di averlo inizialmente ritenuto troppo potente per essere rilasciato pubblicamente, per timore di un utilizzo dannoso [13]. Ma è stata l'applicazione ChatGPT del 2022, con la sua interfaccia utente che riprende il concetto di chatbot, a ricevere un'ampia copertura mediatica e l'attenzione del pubblico [14]. GPT-4 del 2023 è stato elogiato per la sua maggiore accuratezza e definito un "Sacro Graal" per le sue capacità multimodali (i.e.: la sua capacità di analizzare dati di diverso tipo, come testo, audio e immagini) [15]. Il rilascio di ChatGPT ha portato ad un aumento nell'utilizzo degli LLM in diversi sottocampi di ricerca dell'informatica, tra cui robotica ed ingegneria del software, oltre che ad un maggior riguardo per il loro impatto sociale [12]. Nel 2024 OpenAI ha rilasciato il modello di ragionamento OpenAI o1, che genera lunghe catene di pensiero prima di restituire una risposta finale [16]. Negli anni sono stati sviluppati molti LLM con dimensioni paragonabili a quelli della serie GPT di OpenAI [17].
Dal 2022, i modelli linguistici open-source hanno guadagnato popolarità, soprattutto con i primi modelli BLOOM e LLaMA di Meta e anche i modelli Mistral 7B e Mixtral 8x7b di Mistral AI. Nel gennaio 2025, DeepSeek ha rilasciato DeepSeek R1, un modello open-weight da 671 miliardi di parametri che offre prestazioni parago-

nabili a OpenAI o1, ma a un costo molto inferiore [18].
Dal 2023, molti LLM sono stati addestrati per essere multimodali, con la capacità di elaborare o generare anche altri tipi di dati, come immagini o audio. Questi LLM sono anche chiamati grandi modelli multimodali ("Large Multimodal Models") [19].
Ad oggi, i modelli più grandi e performanti sono tutti basati sull'architettura a trasformatore [20].

1.2 Accenni del funzionamento

Fondamentalmente, un Large Language Model (LLM) è in grado di prevedere quale sia la parola più plausibile da inserire alla fine di un testo dato, un processo che in gergo tecnico si chiama inferenza. [21]
Poiché gli algoritmi di Machine Learning possono elaborare esclusivamente informazioni numeriche, il testo in ingresso viene innanzitutto suddiviso in frammenti più piccoli, chiamati *token*. Un token non coincide necessariamente con una parola intera: può rappresentare una parola completa, una parte di essa o anche un singolo carattere, a seconda della suddivisione stabilita dal sistema di tokenizzazione [22].
Ogni token viene poi convertito in un vettore numerico, detto *embedding*. Ogni embedding rappresenta il token in uno spazio a molte dimensioni (spesso centinaia o migliaia), in cui la distanza e la direzione tra vettori riflettono relazioni semantiche e sintattiche tra parole. Ad esempio, gli embedding di "Italia" e "Francia" saranno vicini in quanto entrambe le parole rappresentano nazioni europee. In questo spazio vettoriale si trovano tutti gli embedding dei token che il modello ha incontrato durante la fase di addestramento. È proprio in questa fase che l'algoritmo "impara" a trasformare i token in embedding significativi, modificando gradualmente i propri parametri — talvolta dell'ordine di miliardi — in modo da ridurre l'errore nelle previsioni e aumentare la coerenza dei testi generati [23].
Come accennato prima, i moderni LLM si basano sull'architettura *transformer*, caratterizzata dall'uso del meccanismo dell'attenzione (*attention mechanism*). All'in-

terno di questa architettura, i vari embedding vengono manipolati matematicamente attraverso una serie di passaggi che includono moltiplicazioni di matrici, normalizzazioni e funzioni di attivazione, fino a produrre un vettore finale corrispondente alla previsione di un token specifico. Una volta generato questo token, viene aggiunto alla fine del testo e l'intero contenuto aggiornato viene nuovamente elaborato dal modello, ripetendo il ciclo fino a costruire un testo completo in risposta al prompt dell'utente [24].

Il meccanismo dell'attenzione è un passaggio cruciale: esso permette al modello di valutare, per ogni token in ingresso, quali altri token della sequenza siano più rilevanti per determinarne il significato. In altre parole, non si limita a interpretare un token in maniera isolata, ma lo considera nel contesto più ampio della frase o del paragrafo in cui si trova. Il termine "attenzione" è ispirato a un processo analogo nella cognizione umana, in cui focalizziamo la nostra concentrazione su determinate parti di un testo per comprenderne appieno il senso. Grazie a questa capacità, gli LLM riescono a mantenere coerenza logica anche su sequenze testuali lunghe, collegando correttamente informazioni distanti tra loro e producendo risposte che seguono un filo narrativo o argomentativo esteso [25].

L'introduzione di questo meccanismo ha segnato un punto di svolta nel settore. Tuttavia, la sua applicazione su larga scala è stata possibile solo grazie ai progressi nell'hardware, in particolare all'uso delle GPU (*Graphics Processing Unit*). Le GPU, inizialmente progettate per l'elaborazione grafica, sono straordinariamente efficienti nel calcolo parallelo di operazioni matriciali, proprio quelle su cui si basano il meccanismo dell'attenzione e l'intera architettura transformer [26]. Questo connubio tra innovazione algoritmica e potenza di calcolo ha permesso di addestrare modelli con miliardi di parametri su dataset enormi, aprendo la strada alla generazione di testi complessi e coerenti su scala mai vista prima.

Durante l'addestramento iniziale, il modello viene esposto ad enormi quantità di testo provenienti da fonti eterogenee, imparando a prevedere il token successivo dato un contesto. Per ogni previsione calcola l'errore rispetto al token reale e regola i

propri miliardi di parametri tramite il processo di *backpropagation*, ottimizzandoli gradualmente per ridurre l'errore medio.
Un modello già addestrato può essere ulteriormente specializzato tramite fine-tuning, che consiste nel riaddestrarlo (in parte o per intero) su un set di dati più ristretto e specifico, ad esempio testi tecnici o conversazioni in uno stile particolare. Questa procedura permette di adattare l'LLM a compiti specifici — come assistenza clienti, generazione di codice o analisi di documenti giuridici — senza ripetere da zero il costoso addestramento iniziale [27].
Un'ulteriore tecnica molto diffusa è RLHF (*Reinforcement Learning from Human Feedback*), utilizzata per rendere le risposte del modello più utili, sicure e in linea con i valori umani. In questo approccio, dopo l'addestramento iniziale, il modello genera diverse possibili risposte ad un insieme di prompt; queste risposte vengono valutate da valutatori umani, che le classificano in base a criteri come accuratezza, cortesia o pertinenza. Con queste valutazioni, si addestra un modello a parte capace di sostituire, il più accuratamente possibile, il compito dei valutatori umani. L'LLM viene così ottimizzato per soddisfare al meglio la valutazione di questo modello [28].

2

Model Context Protocol

2.1 Funzionamento

MCP è un protocollo open-source che standardizza il modo in cui le applicazioni forniscono contesto agli LLM. Si può pensare a MCP come ad 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 i modelli IA a diverse fonti di dati e strumenti. MCP consente di creare agenti e flussi di lavoro complessi basati sugli LLM e connette i modelli con il mondo. [29]

Il protocollo MCP include i seguenti progetti:

- Specifiche MCP: una specifica di MCP che delinea 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.

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. [30]

2.1.1 Partecipanti

MCP segue un'architettura client-server in cui un host MCP, un'applicazione IA come Claude Code o Claude Desktop, stabilisce connessioni ad uno o più server MCP. L'host MCP realizza questo creando un client MCP per ogni server MCP. Ogni client MCP mantiene una connessione uno-a-uno dedicata con il suo server MCP corrispondente. I principali partecipanti all'architettura MCP sono:

- Host MCP: l'applicazione IA che coordina e gestisce uno o più client MCP.
- Client MCP: un componente che mantiene una connessione ad un server MCP ed ottiene, da un server MCP, il contesto che l'host MCP può utilizzare.
- Server MCP: un programma che fornisce contesto ai client MCP; possono essere eseguiti sia in locale che in remoto. [30]

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. [30]

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. [30]

Livello di trasporto

Il livello di trasporto gestisce i canali di comunicazione e l'autenticazione tra client e server. Gestisce la 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 con eventi inviati dal server opzionali 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.

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. [30]

2.1.3 Protocollo del livello dati

Una parte fondamentale di MCP è la definizione dello schema e della semantica tra client e server MCP. Il livello dati è la parte di MCP che definisce le modalità con cui gli sviluppatori possono condividere il contesto dai server MCP ai client MCP. 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. [30]

Primitive

Le primitive MCP sono il concetto più importante all'interno di MCP. 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 (ad es.: operazioni su file, chiamate API, query di database).
- Risorse: fonti di dati che forniscono informazioni contestuali alle applicazioni IA (ad es.: contenuto di file, record di database, risposte API).
- Prompt: schemi riutilizzabili che aiutano a strutturare le interazioni con i modelli linguistici (ad es.: prompt di sistema, prompt *few-shot*).

Ogni tipo di primitiva ha metodi associati per la scoperta (`*/list`), il recupero (`*/get`) e, in alcuni casi, l'esecuzione (`tools/call`). I client MCP utilizzeranno

i metodi `*/list` per scoprire le primitive disponibili. Ad esempio, un client può prima elencare tutti gli strumenti disponibili (`tools/list`) e poi eseguirli. Questa progettazione consente di creare elenchi dinamici.
Come esempio concreto, si consideri un server MCP 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 MCP di creare interazioni più ricche.

- Campionamento: consente ai server di richiedere il completamento del modello linguistico dall'applicazione IA del client. Questa funzionalità è utile quando gli autori del server desiderano accedere ad un modello linguistico, ma vogliono rimanere indipendenti dal modello e non includere un SDK del modello linguistico nel proprio server MCP. Possono utilizzare il metodo `sampling/complete` per richiedere il completamento del modello linguistico dall'applicazione IA del client.

- 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` per richiedere informazioni aggiuntive all'utente.

- Logging: consente ai server di inviare messaggi di log ai client a scopo di debug e monitoraggio. [30]

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, come quando vengono rese disponibili nuove funzionalità o vengono modificati

strumenti esistenti, il server può inviare notifiche di aggiornamento per informare i client connessi di tali modifiche. Le notifiche vengono inviate come messaggi di notifica JSON-RPC 2.0 (senza attendere una risposta) e consentono ai server MCP di fornire aggiornamenti in tempo reale ai client connessi. [30]

3

Ticket Management System

Il *Ticket Management System* è un sistema software che ha lo scopo di aiutare gli utenti nella creazione e gestione di ticket, creati per segnalare la presenza di problemi individuati, e nel loro inoltro a degli sviluppatori che possano analizzare e poi risolvere tali problemi. Funzionalità centrale è il fatto che l'utente si interfaccia con un chatbot, nello specifico, un'istanza della risorsa Azure OpenAI che usa il modello GPT-4.1, per usufruire dei servizi offerti del sistema.
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*, 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*.
Il progetto *TM.Shared* definisce tutte le classi degli oggetti che vengono creati internamente nel codice al momento 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` legate, il `TicketStatus` e la `TicketPriority`.

- `Category`: ogni `Ticket` appartiene ad una categoria.
- `Comment`: i `Ticket` hanno una serie di commenti.
- `Task`: i task sono le operazioni affidate 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`.

Il progetto *TM.Data* ha il compito di gestire la persistenza delle informazioni, fornendo l'accesso al database ed archiviando gli oggetti creati.
Il progetto *TM.CQRS* introduce l'implementazione del pattern *Command Query Responsibility Segregation*, un pattern architetturale che separa le operazioni di lettura (`Queries`) da quelle di scrittura (`Commands`) all'interno di un'applicazione.
Il cuore della logica applicativa è rappresentato da *TM.Function*. Essa è un'Azure Function, ovvero una soluzione serverless che consente di scrivere meno codice, gestire un'infrastruttura meno complessa e risparmiare sui costi. Non è più necessario preoccuparsi della distribuzione e della gestione dei server, in quanto l'infrastruttura cloud fornisce tutte le risorse aggiornate necessarie per mantenere le applicazioni in esecuzione [31]. Prende il ruolo dell'MCP Host, l'applicazione IA che gestisce la logica principale del Model Context Protocol. Verrà discusso più approfonditamente in seguito.
Infine, *TM.Client* costituisce l'interfaccia utente, nonchè 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.

3.1 Il Model Context Protocol in pratica

Conclusioni

Ringraziamenti

Desidero esprimere la mia più sincera gratitudine ai docenti, per la dedizione e la passione con cui hanno trasmesso le loro conoscenze, contribuendo in modo fondamentale alla mia crescita accademica; ai colleghi in azienda, per la professionalità e 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.

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 2.

[2] Model Context Protocol. Getting Started - Introduction, 2023. URL: https://modelcontextprotocol.io/docs/getting-started/intro. Citato a pagina 3.

[3] Adam Kilgarriff and Gregory Grefenstette. Introduction to the Special Issue on the Web as Corpus. *Computational Linguistics*, 29(3):333–347, 2003. doi:10.1162/089120103322711569. Citato a pagina 4.

[4] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics, 2001. doi:10.3115/1073012.1073017. Citato a pagina 4.

[5] Philip Resnik and Noah A. Smith. The Web as a Parallel Corpus. *Computational Linguistics*, 29(3):349–380, 2003. Archived from the original on 2024-06-07. Retrieved 2024-06-07. doi:10.1162/089120103322711578. Citato a pagina 4.

[6] Joshua Goodman. A Bit of Progress in Language Modeling. *arXiv preprint*, 2001. URL: https://arxiv.org/abs/cs/0108005, arXiv:cs/0108005. Citato a pagina 4.

[7] Wei Xu and Alex Rudnicky. Can artificial neural networks learn language models? In *6th International Conference on Spoken Language Processing (ICSLP 2000)*, volume 1. ISCA, 2000. doi:10.21437/icslp.2000-50. Citato a pagina 4.

[8] Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sensing*, 13(22):4712, 2021. doi:10.3390/rs13224712. Citato a pagina 4.

[9] 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 5.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint*, 2014. URL: https://arxiv.org/abs/1409.0473, arXiv:1409.0473. Citato a pagina 5.

[11] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020. Archived from the original on 2022-04-03. Retrieved 2024-01-21. arXiv:2002.12327, doi:10.1162/tacl_a_00349. Citato a pagina 5.

[12] Rajiv Movva, Sidhika Balachandar, Kenny Peng, Gabriel Agostini, Nikhil Garg, and Emma Pierson. Topics, Authors, and Institutions in Large Language Model Research: Trends from 17K arXiv Papers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 1223–1243, 2024. Retrieved 2024-12-08. arXiv:2307.10700, doi:10.18653/v1/2024.naacl-long.67. Citato a pagina 5.

[13] Alex Hern. New AI fake text generator may be too dangerous to release, say creators. *The Guardian*, February 2019. Archived from the original on 14 February 2019. Retrieved 20 January 2024. URL: https://web.archive.org/web/20190214173112/https://www.theguardian.com/technology/2019/feb/14/elon-musk-backed-ai-writes-convincing-news-fiction. Citato a pagina 5.

[14] Euronews. ChatGPT a year on: 3 ways the AI chatbot has completely changed the world in 12 months. *Euronews*, November 2023. Archived from the original on January 14, 2024. Retrieved January 20, 2024. URL: https://www.euronews.com/next/2023/11/30/chatgpt-a-year-on-3-ways-the-ai-chatbot-has-completely-changed-the-world-in-12-months. Citato a pagina 5.

[15] Will Heaven. GPT-4 is bigger and better than ChatGPT—but OpenAI won't say why. *MIT Technology Review*, March 2023. Archived from the original on March 17, 2023. Retrieved January 20, 2024. URL: https://web.archive.org/web/20230317224201/https://www.technologyreview.com/2023/03/14/1069823/gpt-4-is-bigger-and-better-chatgpt-openai/. Citato a pagina 5.

[16] Cade Metz. OpenAI Unveils New ChatGPT That Can Reason Through Math and Science. *The New York Times*, September 2024. Retrieved September 12, 2024. URL: https://www.nytimes.com/2024/09/12/technology/openai-chatgpt-math.html. Citato a pagina 5.

[17] Parameters in notable artificial intelligence systems, November 2023. Retrieved January 20, 2024. URL: https://ourworldindata.org/grapher/artificia

l-intelligence-parameter-count?time=2017-09-05..latest. Citato a pagina 5.

[18] Shubham Sharma. Open-source DeepSeek-R1 uses pure reinforcement learning to match OpenAI o1 — at 95% less cost. *VentureBeat*, January 2025. Retrieved 2025-01-26. URL: https://venturebeat.com/ai/open-source-deepseek-r1-uses-pure-reinforcement-learning-to-match-openai-o1-at-95-less-cost/. Citato a pagina 6.

[19] Dr Tehseen Zia. Unveiling of Large Multimodal Models: Shaping the Landscape of Language Models in 2024. *Unite.AI*, January 2024. Retrieved 2024-12-28. URL: https://www.unite.ai/unveiling-of-large-multimodal-models-shaping-the-landscape-of-language-models-in-2024/. Citato a pagina 6.

[20] Rick Merritt. What Is a Transformer Model? NVIDIA Blog, March 2022. Archived from the original on 2023-11-17. Retrieved 2023-07-25. URL: https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/. Citato a pagina 6.

[21] Google Developers. Introduction to Large Language Models, 2024. URL: https://developers.google.com/machine-learning/resources/intro-llms. Citato a pagina 6.

[22] Polo Club. LLM Transformer Model Visually Explained, 2024. URL: https://poloclub.github.io/transformer-explainer/. Citato a pagina 6.

[23] IBM. What are Large Language Models (LLMs)?, 2023. URL: https://www.ibm.com/think/topics/large-language-models. Citato a pagina 6.

[24] Wikipedia contributors. Transformer (deep learning architecture), 2025. URL: https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture). Citato a pagina 7.

[25] IBM. What is an attention mechanism?, 2024. URL: https://www.ibm.com/think/topics/attention-mechanism. Citato a pagina 7.

[26] Wikipedia contributors. Attention Is All You Need, 2025. URL: https://en.wikipedia.org/wiki/Attention_Is_All_You_Need. Citato a pagina 7.

[27] Dave Bergmann (IBM). What is fine-tuning?, 2024. URL: https://www.ibm.com/think/topics/fine-tuning. Citato a pagina 8.

[28] Wikipedia contributors. Reinforcement learning from human feedback, 2025. URL: https://en.wikipedia.org/wiki/Reinforcement_learning_from_human_feedback. Citato a pagina 8.

[29] Model Context Protocol. Getting Started - Introduction, 2023. Accessed: 2025-08-16. URL: https://modelcontextprotocol.io/docs/getting-started/intro. Citato a pagina 9.

[30] Model Context Protocol. Learn - Architecture, 2023. Accessed: 2025-08-16. URL: https://modelcontextprotocol.io/docs/learn/architecture. Citato alle pagine 10, 11, 12, 13, and 14.

[31] Microsoft Docs. Azure Functions overview. Accessed: 2025-08-16. URL: https://learn.microsoft.com/it-it/azure/azure-functions/functions-overview?pivots=programming-language-csharp. Citato a pagina 16.