



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Utilizzo di Modelli Linguistici Avanzati per l'Interrogazione di Dati Sanitari

RELATORE

Prof. Gennaro Costagliola

CANDIDATO

Teodoro Grauso

Matricola: 0512111084

*”Per tutto ciò che oggi è meglio di quanto avrei
mai potuto immaginare...”*

Abstract

L'uso dei Large Language Model per la generazione di un testo basato sulle risposte API di un server FHIR rappresenta una soluzione all'avanguardia nel settore sanitario. Il processo inizia con la creazione di una richiesta API utilizzando le capacità generative di GPT-4o. Dopo aver ottenuto la risposta dal server FHIR, i dati vengono analizzati e trasformati in una rappresentazione testuale attraverso i modelli locali di Large Language Model. Questi modelli elaborano le informazioni e producono descrizioni coerenti.

Il vantaggio principale di questo approccio è la capacità di creare descrizioni precise e pertinenti per i medici, permettendo loro di comprendere rapidamente le informazioni cliniche senza dover esaminare direttamente i dati grezzi provenienti dal server. Questo processo migliora l'efficienza del flusso di lavoro medico, riduce il rischio di fraintendimenti e favorisce una comunicazione più efficace tra gli operatori sanitari, nel pieno rispetto della privacy dei pazienti.

Indice

1	Introduzione	1
1.1	Struttura Tesi1
1.2	L’Intelligenza Artificiale2
1.3	I Large Language Model2
1.4	Fast Healthcare Interoperability Resources3
1.5	L’Intelligenza Artificiale nella Sanità3
1.6	Esecuzione di Large Language Model5
2	Framework di Orchestrazione per LLM	7
2.1	L’Orchestrazione7
2.2	API per l’inferenza dei modelli8
2.2.1	GPT-4all Python8
2.2.2	LLaMa.cpp9
2.3	Framework di Orchestrazione per i LLM9
2.4	Haystack10
2.4.1	Componenti11
2.4.2	Pipeline12
2.4.3	Agenti12
2.5	LlamaIndex12
2.5.1	Index13
2.5.2	Retriever14
2.5.3	Query Engine14
2.6	LangChain14
2.6.1	Model I/O15
2.6.2	Retrieval16
2.6.3	Agents17
3	I Large Language Model	21
3.1	Applicazioni dei LLM21

3.2	Training dei Modelli	22
3.3	Valutazione dei Modelli	24
3.3.1	Il Punteggio ROUGE	25
3.3.2	Altre Metriche	26
3.4	GPT4All vs LLaMa	27
4	Implementazione	29
4.1	Analisi e Raccolta dei Requisiti	29
4.1.1	Scopo del sistema	30
4.1.2	Ambito del sistema	30
4.1.3	Obiettivo e Criteri di Successo del Sistema	31
4.1.4	Analisi dei sistemi correnti	31
4.1.5	Il Sistema Proposto	34
4.2	Fase di progettazione di Medical-AI	35
4.2.1	Gli obiettivi del sistema	36
4.2.2	Architettura del Sistema	36
4.3	Lo sviluppo di Medical-AI	37
4.3.1	API di OpenAI	38
4.3.2	Utilizzo di LangChain	39
4.3.3	Prompt Engineering	43
4.4	Esempio di Esecuzione di Medical-AI	46
4.5	Manuale Utente	47
4.5.1	Accesso a Medical-AI	47
4.5.2	Profilo Utente	49
4.5.3	Accesso all'Assistente Virtuale	49
4.6	Valutazione dei Risultati	53
5	Conclusioni	65

Capitolo 1

Introduzione

L’evoluzione delle tecnologie nell’ambito della gestione delle informazioni sanitarie ha rivoluzionato il modo in cui accediamo, archiviamo e interpretiamo i dati clinici. In questo contesto, il Framework for the Interoperability of Health Information Resources (FHIR) [14], si è affermato come uno standard emergente per la trasmissione e lo scambio di informazioni sanitarie, offrendo una struttura flessibile basata su risorse in formato JavaScript Object Notation (JSON). Mentre la standardizzazione dei dati promuove l’interoperabilità tra sistemi, la comprensione e l’utilizzo efficiente delle informazioni contenute in queste risposte rimangono una sfida.

La presente tesi si propone di affrontare questa sfida esplorando l’applicazione dei Large Language Models (LLM)) per la rielaborazione delle risposte JSON fornite dai server FHIR. L’obiettivo primario è tradurre in linguaggio naturale le informazioni strutturate, consentendo una più facile interpretazione e utilizzo da parte degli operatori sanitari. I LLM, basati su avanzate reti neurali, si sono dimostrati strumenti straordinariamente efficaci nella generazione di testo coerente e significativo. Intendiamo sfruttare questa capacità per trasformare dati complessi in narrazioni comprensibili, facilitando così la comunicazione e migliorando la fruibilità delle informazioni cliniche.

1.1 Struttura Tesi

Il seguente lavoro di tesi è così organizzato: nel secondo capitolo viene fornita una panoramica sui framework di orchestrazione, analizzando nello specifico quelli che orchestrano i processi di interazione con i LLM. Nel terzo capitolo viene riportata una descrizione completa dei Large Language Model, spiegando cosa sono, e il loro funzionamento. Il quarto capitolo viene riportata una descrizione completa delle fasi di sviluppo del software Medical-AI, con una successiva analisi dei prompt utilizzati per interagire con i LLM. Infine, il quinto capitolo mostra le conclusioni relative allo studio fatto.

1.2 L’Intelligenza Artificiale

L’intelligenza artificiale è una branca dell’informatica che mira a creare sistemi in grado di eseguire compiti che normalmente richiedono l’intelligenza umana, come apprendere da esperienze passate, adattarsi a nuove situazioni e risolvere problemi complessi. L’obiettivo fondamentale dell’intelligenza artificiale è emulare la capacità cognitiva umana, permettendo alle macchine di eseguire attività intelligenti attraverso l’apprendimento automatico, l’elaborazione del linguaggio naturale e la visione artificiale.

La ricerca nell’ambito dell’AI ha raggiunto progressi significativi, grazie anche all’impiego di algoritmi avanzati e all’incremento della potenza di calcolo disponibile nei sistemi moderni. L’apprendimento automatico, in particolare, ha consentito alle macchine di migliorare le loro prestazioni attraverso l’analisi dei dati e l’identificazione di modelli senza una programmazione esplicita. Le applicazioni dell’AI sono diversificate e comprendono settori quali la medicina, l’industria, l’automazione, la finanza e molti altri.

Un aspetto rilevante dell’AI sono i Large Language Models (Modello Linguistico di Grandi Dimensioni), che rappresentano una categoria di algoritmi di apprendimento automatico in grado di comprendere e generare testo in modo avanzato.

1.3 I Large Language Model

I Large Language Models sono modelli di intelligenza artificiale addestrati su grandi quantità di dati testuali provenienti da diverse fonti. Uno degli esempi più noti di tali modelli è GPT-4o (Generative Pre-trained Transformer), sviluppato da OpenAI [23]. GPT-4o è un modello di linguaggio che contiene 20 miliardi di parametri rendendolo uno dei più grandi al mondo.

Questi modelli sono ”pre-addestrati” su una vasta gamma di testi, il che significa che imparano la struttura della lingua e le relazioni semantiche presenti nei dati di addestramento. Successivamente, possono essere ”raffinati” per eseguire specifiche attività o compiti.

Un aspetto notevole di questi modelli è la loro capacità di generare testo coerente e contestualmente rilevante. Possono essere utilizzati per rispondere a domande, completare frasi, generare contenuti creativi e altro ancora. È importante sottolineare che, nonostante le loro impressionanti capacità, i Large Language Models non hanno una vera comprensione nel senso umano e possono occasionalmente produrre risposte errate.

1.4 Fast Healthcare Interoperability Resources

Il Framework for the Interoperability of Health Information Resources rappresenta uno standard avanzato nel panorama delle Informazioni Sanitarie Elettroniche (EHR) e nella gestione dei dati clinici. Questo framework è stato sviluppato dall'organizzazione HL7 International, un'autorità riconosciuta nello sviluppo di standard sanitari globali, con l'obiettivo di migliorare la capacità di interoperabilità tra i vari sistemi informativi nel settore della salute.

La caratteristica distintiva di FHIR risiede nella sua progettazione agile e nell'utilizzo di moderni standard web, che facilitano la trasmissione efficace e sicura dei dati clinici. FHIR mira a semplificare e rendere più efficiente la condivisione delle informazioni sanitarie, consentendo una comunicazione più fluida tra diverse piattaforme e applicazioni nel contesto sanitario.

1.5 L'Intelligenza Artificiale nella Sanità

L'Intelligenza Artificiale (AI) ha rivoluzionato il settore sanitario introducendo soluzioni innovative per migliorare l'accesso e l'efficacia dell'assistenza medica. Diversi articoli evidenziano le possibili applicazioni dell'AI nella sanità, dal supporto alla diagnosi e al trattamento delle malattie fino alla gestione dei dati e all'ottimizzazione dei processi.

Gli strumenti di intelligenza artificiale generativa, tra cui il modello GPT-4 e l'interfaccia di chatbot, stanno aprendo nuove e promettenti prospettive in una vasta gamma di applicazioni nell'ambito sanitario.

Ad esempio l'uso dell'Intelligenza Artificiale nell'industria farmaceutica ha registrato una crescita significativa, con l'obiettivo di migliorare la scoperta dei farmaci, ridurre i costi di ricerca e sviluppo, diminuire i fallimenti dei trial clinici e per la produzione di farmaci più efficaci. Alcune partnership notevoli, come quelle tra Pfizer e IBM Watson, Sanofi Genzyme e Recursion Pharmaceuticals, testimoniano il crescente interesse in materia.

Nonostante l'inevitabile impatto dell'AI nel campo dell'assistenza sanitaria, è importante sottolineare che attualmente non esistono farmaci basati su AI che abbiano ricevuto l'approvazione della Food and Drug Administration. Questa mancanza di approvazione deriva dal fatto che l'AI non può sostituire integralmente i processi sperimentali tradizionali nel contesto della ricerca farmaceutica. Ciononostante, è fondamentale riconoscere il ruolo significativo che l'AI può svolgere nel catalizzare progressi nella scoperta e nello sviluppo di nuovi farmaci attraverso l'analisi avanzata di dati genomici e proteomici. L'AI si dimostra particolarmente efficace nell'interpretare e analizzare enormi quantità di dati biologici, consentendo agli scienziati di identificare correlazioni complesse e modelli

nascosti. Attraverso l'applicazione di algoritmi avanzati, l'AI può accelerare il processo di identificazione dei bersagli terapeutici e la progettazione di composti potenziali, riducendo notevolmente i tempi di ricerca [1].

Il progresso dell'interoperabilità sanitaria riveste un'importanza cruciale per la ricerca medica. Le collaborazioni tra agenzie federali come ONC, CDC e CMS mirano all'adozione dello standard FHIR per la digitalizzazione delle cartelle cliniche. La trasformazione dei dati EHR in risorse FHIR presenta delle sfide. L'articolo in questione [17] valuta l'efficacia di FHIR-GPT, un modello linguistico di grandi dimensioni, nel convertire testi clinici in risorse FHIR. Su 3.671 frammenti di testo, FHIR-GPT mostra un tasso di corrispondenza esatta oltre il 90%, superando le metodologie esistenti. I risultati mostrano il potenziale dei modelli linguistici nel migliorare l'interoperabilità dei dati sanitari, con prospettive future di estensione a risorse FHIR aggiuntive.

Gli assistenti virtuali nel contesto sanitario svolgono un ruolo fondamentale nel fornire una vasta gamma di servizi, strumenti e interazioni rivolte a diversi utenti, tra cui pazienti, operatori sanitari e professionisti del settore. Questi assistenti sono progettati per offrire consulenza, supporto e gestione di diverse patologie, contribuendo così a migliorare l'esperienza complessiva nell'ambito della salute. In particolare, i pazienti, soprattutto coloro che affrontano malattie croniche che richiedono l'assunzione regolare di farmaci, possono trovarsi di fronte a difficoltà nel ricordare le dosi corrette. Questa situazione potrebbe avere un impatto negativo sull'efficacia complessiva del trattamento. Gli assistenti virtuali permettono di affrontare queste sfide offrendo funzionalità di promemoria e monitoraggio dell'assunzione dei farmaci. Possono inviare notifiche personalizzate ai pazienti per ricordare loro di prendere i farmaci nei tempi prestabiliti e tenere traccia dell'aderenza al trattamento nel tempo [18].

Nel contesto delle malattie croniche, è stato dimostrato che i pazienti possono trarre notevoli vantaggi dall'utilizzo di chatbot progettati per monitorare le loro condizioni, fornire informazioni specifiche e promuovere l'aderenza ai trattamenti farmacologici. In un recente studio, è stata presentata un'architettura di chatbot concepita per fornire assistenza ai pazienti affetti da malattie croniche. Tale architettura si basa su tre pilastri fondamentali: la scalabilità attraverso l'implementazione di microserivizi, l'adozione di modelli standard per la condivisione dei dati mediante FHIR e l'impiego di un approccio normativo per la modellazione delle conversazioni tramite Artificial Intelligence Markup Language (AIML). Questa proposta offre una base strutturata e robusta per lo sviluppo di chatbot mirati all'assistenza personalizzata dei pazienti cronici, evidenziando l'importanza di un'architettura software ben definita per ottimizzare l'efficacia di tali sistemi nell'ambito sanitario [25].

Sebbene esistano numerosi "assistenti virtuali" rivolti ai pazienti, vi è un vuoto per

quanto riguarda il supporto al personale medico. L'integrazione di un modello linguistico generativo, l'adozione degli standard FHIR per la gestione dei dati e l'approccio alla sicurezza tramite un LLM locale forniscono una soluzione alle sfide presenti nel contesto dell'assistenza virtuale in ambito medico. Questo chatbot si propone come un vero e proprio assistente per il personale medico, contribuendo nella gestione quotidiana dei pazienti. In linea con il progresso nell'ambito sanitario, dove l'intelligenza artificiale ha introdotto soluzioni innovative per migliorare l'accesso e l'efficacia dell'assistenza medica, l'utilizzo di strumenti come il modello GPT-4 e un'interfaccia di chatbot offre nuove prospettive nel campo della sanità.

1.6 Esecuzione di Large Language Model

LangChain è un framework di orchestrazione open source progettato per agevolare lo sviluppo di applicazioni che sfruttano i Large Language Models.

Questo strumento svolge il ruolo di un'interfaccia generica per una vasta gamma di LLM, offrendo un ambiente centralizzato per la creazione di applicazioni basate su questi modelli e per l'integrazione con fonti di dati esterne e flussi di lavoro software. L'approccio modulare di LangChain consente agli sviluppatori e ai data scientist di confrontare dinamicamente diversi prompt e modelli di fondazione, senza la necessità di riscrivere il codice.

LangChain semplifica numerosi casi d'uso per i Large Language Models e il Natural Language Processing (NLP), tra cui chatbot, ricerca intelligente, risposta alle domande, servizi di riepilogo e agenti virtuali in grado di implementare la Robotic Process Automation (RPA).

Capitolo 2

Framework di Orchestrazione per LLM

L'orchestrazione dei processi rappresenta un elemento chiave nella gestione dei modelli linguistici di grandi dimensioni. I framework di orchestrazione per gli LLM sono strumenti essenziali per gestire l'incremento di potenza e di complessità degli attuali modelli. Questi strumenti semplificano la pianificazione delle risorse, il carico di lavoro e la gestione degli errori, garantendo efficienza, affidabilità, scalabilità e sicurezza.

Sorgono però anche nuove sfide nell'utilizzo di tali framework, data la loro complessità e il loro costo di implementazione e manutenzione. Tra questi, si distinguono Haystack, LlamaIndex e LangChain per la loro facilità d'uso, flessibilità ed efficienza. Essi forniscono un'interfaccia intuitiva e una vasta gamma di funzionalità per sfruttare appieno i modelli LLM moderni.

2.1 L'Orchestrazione

L'orchestrazione dei processi si riferisce all'unificazione di singole attività in processi completi e alla semplificazione delle integrazioni fra sistemi con connettori universali, integrazioni dirette o adattatori di API .

Il suo obiettivo è concatenare diverse attività per eseguire flussi di lavoro o processi più ampi. Questi processi coinvolgono molteplici attività automatizzate e sistemi diversi. L'orchestrazione semplifica l'esecuzione dei processi frequenti e ripetibili, aiutando i team a gestire compiti e flussi di lavoro complessi. Quando un processo è ripetibile e le attività possono essere automatizzate, l'orchestrazione consente di risparmiare tempo, aumentare l'efficienza ed eliminare ridondanze [12].

2.2 API per l'inferenza dei modelli

Nella ricerca e nello sviluppo di sistemi basati sui Large Language Models, l'efficace gestione delle API per l'inferenza locale riveste un ruolo cruciale. Queste API consentono di utilizzare e integrare modelli basati sull'architettura LLAMA di Meta e l'ecosistema gpt-4all all'interno di applicazioni locali, garantendo prestazioni ottimali e una maggiore flessibilità operativa.

2.2.1 GPT-4all Python

Le API open-source di GPT-4all permettono a tutti di creare e distribuire modelli linguistici di grandi dimensioni personalizzati e potenti, in grado di essere eseguiti su hardware di device consumer [3].

La caratteristica distintiva di GPT-4all è la privacy. I dati e le query (prompt) vengono gestiti localmente, evitando server esterni e la condivisione di informazioni con terzi. Questo aspetto lo rende particolarmente interessante in attività in cui si desidera proteggere la riservatezza dei propri dati [5], come nel nostro caso: la gestione dei dati sanitari degli utenti.

GPT-4all offre molteplici altri vantaggi. La portabilità è un punto chiave, i suoi modelli richiedono da quattro a otto gigabyte di memoria e possono essere facilmente trasferiti. Questo significa che è possibile utilizzare GPT-4all su praticamente qualsiasi computer moderno senza dover preoccuparti di requisiti hardware o di una GPU dedicata.

Inoltre, mentre molti modelli richiedono una connessione Internet costante, GPT-4all consente di continuare a utilizzare il servizio anche in assenza di una connessione attiva. Tutti i dati necessari sono già presenti nel pacchetto scaricato, consentendo di avviare una conversazione, oppure personalizzare i modelli, senza dipendere da una connessione Internet stabile.

Infine, il fatto che i diversi modelli forniti siano distribuiti con licenza GPL-2 apre diverse opportunità. Questo tipo di licenza consente a chiunque di personalizzare e integrare i modelli per utilizzi commerciali senza dover pagare una licenza. Questa flessibilità rende GPT-4all non solo conveniente, ma anche accessibile e adatto a una varietà di esigenze e progetti.

L'ampia gamma di campi di applicazione offerte da GPT-4all lo rende un prezioso strumento in diversi ambiti. Dal supporto alla scrittura o alla generazione di codice, dalla creatività artistica alla traduzione automatica, questo framework si presenta come un'opportunità versatile per svariati compiti. La sua capacità di generare testo creativo, rispondere a domande tecniche complesse e persino creare chatbot personalizzati offre un ventaglio di possibilità praticamente illimitato.

2.2.2 LLaMa.cpp

LLaMa.cpp, sviluppata da Georgi Gerganov, è una libreria in C++ che implementa l'architettura LLaMa di Meta. Si propone come un'alternativa più leggera e portabile rispetto ai framework esistenti per l'inferenza dei modelli LLM [11].

La sua progettazione come libreria CPU-first semplifica l'integrazione in differenti ambienti di programmazione e ne consente l'adozione su diverse piattaforme fornendo supporto nativo per Linux, MacOS, Windows. Il suo obiettivo principale è consentire l'inferenza dei modelli LLM con un setup minimale e prestazioni all'avanguardia su una vasta gamma di hardware, sia localmente che nel cloud. Inoltre LLaMa.cpp supporta backend di accelerazione hardware personalizzati per il calcolo parallelo, sfruttando le GPU tramite CUDA, ROCm, OpenCL e Metal, fornendo un'esecuzione efficiente dei modelli LLaMa su diverse piattaforme.

Una caratteristica rilevante è la possibilità di avere una quantizzazione personalizzata, che consente un'efficiente moltiplicazione di matrici a bassa precisione; ciò riduce significativamente la larghezza di banda della memoria mantenendo la precisione nelle previsioni del modello. L'uso efficiente delle risorse da parte della libreria garantisce che i modelli linguistici possano essere implementati con un impatto minimo sulla memoria, un fattore cruciale negli ambienti di produzione.

Inoltre, LLaMa.cpp implementa il multi-threading e l'elaborazione batch, che consentono una generazione di token estremamente paralleli tra i core della CPU, contribuendo a un'inferenza del modello più rapida e reattiva.

In particolare consideriamo llama-cpp-python, una Python Bindings che fornisce un'interfaccia per accedere alle API di llama.cpp, consentendone l'utilizzo delle sue funzionalità di base direttamente tramite codice Python [16] [10].

2.3 Framework di Orchestrazione per i LLM

I modelli linguistici di grandi dimensioni, come Gemini di Google e GPT-4 di OpenAI, stanno continuamente aumentando la loro potenza e complessità. Questo rende la gestione e l'orchestrazione di tali modelli un compito sempre più impegnativo. I framework di orchestrazione per LLM sono strumenti software progettati per semplificare la gestione e l'esecuzione di questi modelli.

Utilizzare i framework di orchestrazione per LLM offre vari vantaggi:

- **Efficienza:** Ottimizzano l'uso delle risorse, riducendo i costi operativi, consentendo di stimare e organizzare le risorse computazionali e di memoria necessarie per l'esecuzione dei LLM

-
- Affidabilità: Assicurano che i LLM siano sempre disponibili e funzionanti quando necessario, gestendo il carico di lavoro distribuendolo tra più server per ottimizzare le prestazioni. Facilitano inoltre la gestione degli errori e il ripristino del servizio in caso di problemi, contribuendo a mantenere il sistema operativo in modo efficiente e affidabile.
 - Scalabilità: Consentono di scalare i LLM in modo rapido e efficiente per soddisfare le esigenze in continua evoluzione.
 - Sicurezza: Possono contribuire a proteggere i LLM da accessi non autorizzati e da attacchi informatici.

Eppure ci sono alcune sfide da affrontare nell'utilizzo di questi framework, dato che possono essere complessi da configurare e gestire. La loro implementazione e la manutenzione può essere costosa e richiedono competenze specifiche che potrebbero non essere facilmente reperibili internamente al team di sviluppo.

2.4 Haystack

Haystack è un framework end-to-end che offre strumenti completi per lo sviluppo di sistemi di elaborazione del linguaggio naturale che utilizzano i LLM e modelli Transformer. Con Haystack è possibile utilizzare vari modelli, ospitati su piattaforme come Hugging Face, OpenAI, Cohere o anche modelli distribuiti su SageMaker e i modelli locali personalizzati.



Figura 2.1: Model Providers. Fonte: haystack.deepset.ai

Haystack offre un'ampia gamma di possibilità per la creazione di sistemi avanzati basati su linguaggio naturale:

- Condurre ricerche semantiche su collezioni di documenti di dimensioni considerevoli, indipendentemente dalla lingua in cui sono scritti. Ciò significa che è possibile ottenere risultati pertinenti e significativi anche in contesti multilingue.
- Sviluppare sistemi in grado di generare risposte automatiche e generative a domande poste su basi di conoscenza che contengono una varietà di tipi di informazioni, tra cui testo, immagini e dati tabellari. Questa flessibilità permette di gestire conoscenze eterogenee e di fornire risposte complete e pertinenti agli utenti.
- Creazione di chatbot in linguaggio naturale, sfruttando modelli generativi. Con Haystack, si possono sviluppare chatbot che interagiscono in modo naturale con gli utenti, comprendendo e rispondendo alle loro richieste in modo coerente e preciso.
- Implementare agenti in grado di risolvere query complesse, utilizzando modelli di grandi dimensioni per comprendere e analizzare in profondità le richieste degli utenti. Questo permette di affrontare scenari e domande più complesse, offrendo risposte dettagliate e accurate.
- Estrarre informazioni rilevanti dai documenti al fine di popolare database o costruire grafici. Questo processo di estrazione consente di sfruttare al massimo i dati contenuti nei documenti, trasformandoli in conoscenze utili e accessibili per ulteriori analisi e applicazioni.

Haystack è costituito da diversi elementi: componenti, pipeline e agenti. [24].

2.4.1 Componenti

Il nucleo di Haystack sono i suoi componenti, elementi capaci di svolgere svariate attività come il recupero e il riepilogo di documenti e la generazione di testo. Ogni singolo componente è in grado di gestire modelli linguistici locali o comunicare con un modello ospitato tramite un'API. Haystack offre una serie di componenti pronti all'uso, ma permette anche la creazione dei propri componenti personalizzati. È possibile concatenare i componenti per creare pipeline, che costituiscono l'architettura base delle applicazioni NLP in Haystack. [24]

2.4.2 Pipeline

Le pipeline sono strutture composte da componenti come il Retriever e il Reader, collegati a elementi di infrastruttura come il DocumentStore (ad esempio, Elasticsearch o Weaviate) per formare sistemi complessi.

Haystack offre pipeline già pronte per le attività più comuni, come la risposta a domande effettuate dall'utente, il recupero di documenti o il riepilogo degli stessi. Ma è altrettanto semplice progettare e creare una pipeline personalizzata per scenari NLP molto più complessi [24].

2.4.3 Agenti

L'agente in Haystack utilizza un LLM per risolvere compiti complessi. Durante l'inizializzazione dell'agente, vengono forniti una serie di strumenti, che possono essere sia componenti della pipeline che intere pipeline stesse. L'agente può utilizzare questi strumenti in modo iterativo per arrivare a una risposta.

Quando viene posta una domanda, l'agente determina quali strumenti sono utili per rispondere e li chiama in loop fino a quando non ottiene una risposta. In questo modo, è in grado di ottenere molto di più rispetto alle semplici pipeline di risposta alle domande, sia estrattive che generative [24].

2.5 LlamaIndex

LlamaIndex è un framework per applicazioni basate sui LLM che beneficiano dell'ampliamento del contesto. LlamaIndex fornisce le astrazioni essenziali per acquisire, strutturare e accedere più facilmente a dati privati, o specifici del dominio, al fine di inserirli in modo sicuro e affidabile nei LLM per una generazione di testo più accurata.

Una delle caratteristiche distintive di LlamaIndex è il suo approccio al contesto arricchito attraverso il metodo chiamato Retrieval-Augmented Generation (RAG). Questa metodologia, invece di limitarsi a un tradizionale addestramento di modelli su dati specifici, recupera attivamente informazioni dalle sorgenti di dati esistenti e le incorpora nel contesto della domanda.

Questo processo supera le limitazioni del fine-tuning tradizionale, garantendo che il testo generato sia non solo accurato ma anche aggiornato, rispondendo così alle esigenze di un ambiente in continua evoluzione.

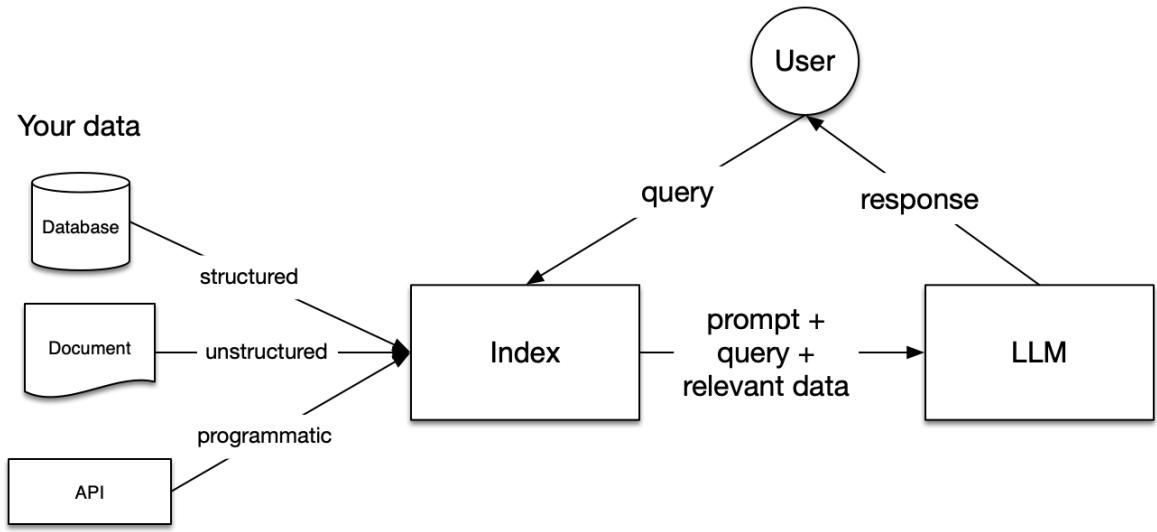


Figura 2.2: Basic RAG. Fonte: llamaindex.ai

LlamaIndex fornisce una serie di strumenti essenziali per la gestione dei dati, tra cui connettori dati, che consentono l'importazione agevole dei dati da una vasta gamma di fonti, come API, PDF e database SQL. Questi connettori sono fondamentali per la creazione di un ecosistema dati completo e diversificato, indispensabile per supportare i processi decisionali basati sull'analisi dei dati.

Inoltre, LlamaIndex si distingue per la sua versatilità. Non impone restrizioni sull'utilizzo dei modelli di linguaggio, consentendo agli utenti di sfruttarli in modi diversi, come completamento automatico, chatbot o agenti semi-autonomi. LlamaIndex agisce come un catalizzatore, migliorando la pertinenza e l'utilità dei modelli di linguaggio esistenti, adattandoli alle esigenze specifiche degli utenti e dei loro dati.

LlamaIndex è un sistema sofisticato composto da tre componenti chiave che costituiscono la sua struttura portante, permettendo l'estrazione di informazioni dai dati o dai documenti:

LlamaIndex è composto da tre componenti principali che costituiscono la sua struttura portante per estrarre informazioni dai dati o dai documenti [20].

2.5.1 Index

Questa è una struttura dati essenziale all'interno di LlamaIndex. Gli Index agiscono come un meccanismo per recuperare rapidamente informazioni rilevanti dai documenti esterni, in risposta alle query dell'utente. Sono fondamentali per arricchire il contesto delle risposte, garantendo che l'informazione estratta sia pertinente e tempestiva [20].

2.5.2 Retriever

Questo componente svolge un ruolo vitale nel reperimento dei dati. È progettato per estrarre e raccogliere informazioni pertinenti in base alle richieste degli utenti. Il Retriever opera in stretta sinergia con l'indice, garantendo un accesso efficiente e accurato alle informazioni desiderate [20].

2.5.3 Query Engine

Il Query Engine è costruito sopra l'infrastruttura dell'Index e del Retriever, il Query Engine offre un'interfaccia universale per interrogare i dati. È responsabile della gestione delle richieste di informazioni, garantendo che gli utenti possano accedere alle risorse necessarie in modo rapido ed efficiente [20].

2.6 LangChain

LangChain [7] è un framework all'avanguardia nel campo dell'intelligenza artificiale. Questa piattaforma coordina l'interazione tra modelli e altre fonti di dati, agevolando lo sviluppo di applicazioni basate sul linguaggio naturale, come chatbot e assistenti virtuali. Queste applicazioni sono in grado di comprendere e generare testo in modo intelligente, migliorando l'interazione utente e automatizzando processi complessi.

Grazie alla sua capacità di gestire e organizzare risorse come dizionari e dataset, LangChain semplifica notevolmente il processo di sviluppo e manutenzione delle applicazioni basate sul linguaggio naturale, inoltre offre una solida base per personalizzare e adattare i modelli linguistici alle specifiche esigenze di progetto.

LangChain si basa sul concetto di "catene", principi fondamentali che organizzano i vari componenti dell'intelligenza artificiale per generare risposte contestualmente consapevoli. Una catena in LangChain rappresenta una sequenza di azioni automatiche che trasformano l'input dell'utente nell'output desiderato.

Le catene sono composte da "collegamenti", ciascuno dei quali rappresenta un'azione specifica all'interno del processo. I collegamenti consentono di suddividere attività complesse in passaggi più gestibili. Alcuni esempi di collegamenti includono la formattazione dell'input dell'utente, l'invio di query a modelli linguistici, il recupero di dati da archivi cloud e la traduzione tra lingue diverse.

Inoltre il framework è costituito da diversi moduli: Model I/O, Retrieval, Agents [7].

2.6.1 Model I/O

L'elemento centrale di un'applicazione che utilizza i modelli linguistici, e il framework LangChain, è il Modello I/O.

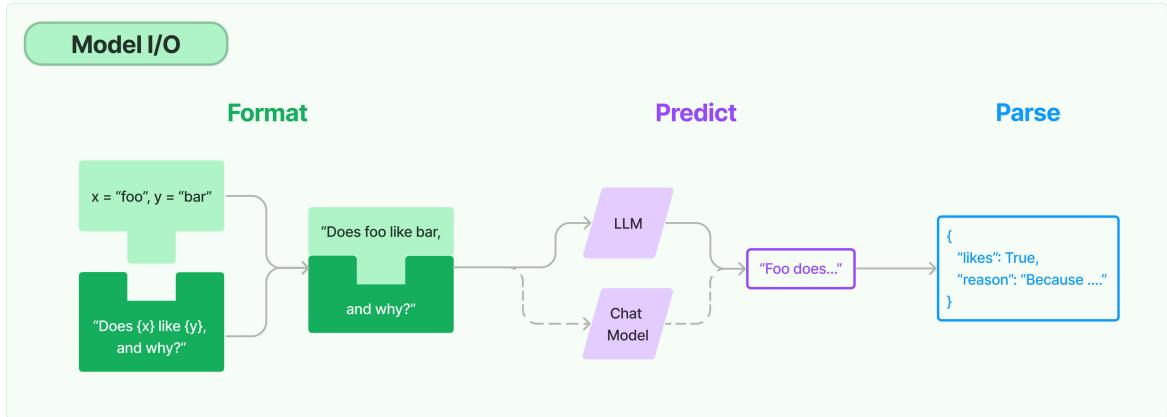


Figura 2.3: Il Modello I/O di LangChain. Fonte: langchain.com

Ci sono due tipi principali di modelli con cui LangChain si integra: gli LLMs e i Chat Model, distinti in base ai tipi di input e output. Gli LLM in LangChain sono modelli di completamento di testo puro. Le API che li utilizzano richiedono come input una stringa e restituiscono una stringa completata, in base al prompt fornito. I Chat Model, d'altra parte, sono simili agli LLM ma ottimizzati nel caso specifico per le conversazioni. Anche se spesso sono formati da LLM, utilizzano un'interfaccia diversa. Invece di una singola stringa, prendono un elenco di messaggi come input e restituiscono un messaggio come output [7].

Messages

Esistono diversi tipi di messaggi, e tutti hanno un **role** e un **content**. Il campo **role** identifica chi ha scritto il messaggio, mentre il campo **content** descrive il contenuto del messaggio e può assumere diverse forme:

- Una stringa (la maggior parte dei modelli opera in questo modo)
- Un elenco di dizionari (utilizzato per l'input multimodale). Ogni dizionario contiene informazioni su quel tipo di input e la sua posizione

I tipi di messaggi sono così divisi:

- HumanMessage: rappresenta un messaggio da parte dell'utente.

-
- AIMessage: rappresenta un messaggio restituito dal modello utilizzato.
 - SystemMessage: rappresenta un messaggio di sistema che indica al modello come comportarsi.

Inoltre, i messaggi hanno una proprietà denominata `additional_kwargs`. Qui possono essere inserite informazioni aggiuntive sui messaggi, ad esempio parametri di input specifici del provider e non generali [7].

Prompts

Gli input dei modelli linguistici sono spesso chiamati prompt. L'input dell'utente non è l'input diretto al modello. Piuttosto, esso viene trasformato in qualche modo per produrre la stringa finale da inserire nel modello. Questa trasformazione è eseguita dai "modelli di prompt" [7].

Parser di output

Nel contesto dei modelli linguistici, l'output può presentarsi sotto forma di stringhe o messaggi. Questi output contengono informazioni strutturate che devono essere elaborate ulteriormente per renderle utilizzabili. I parser di output sono strumenti che svolgono questo compito, trasformando l'output grezzo dei modelli in una forma più gestibile.

Uno dei parser più semplici è il `StrOutputParser`, il cui compito è convertire l'output di un modello linguistico in una stringa. Se l'output è una semplice stringa, il parser la restituisce senza modificarla. Se invece l'output è un messaggio strutturato, come nel caso di un ChatModel, il parser estrae il contenuto dal campo `content` del messaggio e lo restituisce sotto forma di stringa [7].

2.6.2 Retrieval

Molte applicazioni che utilizzano gli LLM richiedono dati specifici dell'utente che non fanno parte del set di dati di addestramento del modello. La metodologia principale per realizzare ciò è attraverso la Retrieval Augmented Generation (RAG): i dati esterni vengono recuperati e poi passati al LLM durante la fase di generazione. LangChain fornisce tutti i blocchi per costruire applicazioni RAG, da quelle più semplici ad altre più complesse.

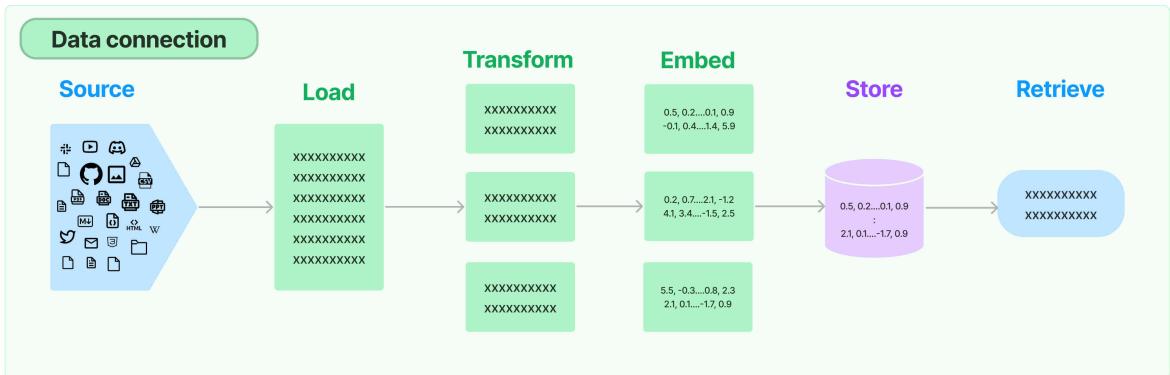


Figura 2.4: Data Connection. Fonte: langchain.com

LangChain offre una vasta gamma di document loaders (oltre 100) in grado di recuperare documenti da fonti diversificate facilitando il caricamento di file nei formati principali (HTML, PDF e codice) da diverse piattaforme.

Un aspetto cruciale del recupero dei documenti è l'abilità di estrarre le informazioni più rilevanti. Questo processo richiede diverse fasi di trasformazione per preparare i documenti per il recupero. La prima fase è la suddivisione, o chunking, di documenti di grandi dimensioni in porzioni più gestibili. LangChain mette a disposizione diversi algoritmi di trasformazione per eseguire questa operazione, insieme a logiche ottimizzate per tipi specifici di documenti, come codice sorgente o markdown.

Un'ulteriore fase nel processo di recupero dei documenti è la creazione di embedding per i testi. Essi catturano il significato semantico del testo, rendendo possibile individuare rapidamente e efficientemente altre parti di testo simili. LangChain offre integrazioni con oltre 25 fornitori diversi di embedding, che vanno da soluzioni open-source a API proprietarie, consentendo agli utenti di scegliere la soluzione più adatta alle proprie esigenze. Inoltre fornisce un'interfaccia standard che semplifica il passaggio da un modello all'altro con facilità [7].

2.6.3 Agents

Gli agenti impiegano un modello linguistico per selezionare una sequenza di azioni da eseguire. Negli agenti, un modello linguistico funge da motore di ragionamento per determinare quali azioni intraprendere e in quale sequenza devono essere eseguite. Esistono diversi tipi di agenti, classificati in base a:

1. Destinazione dell'Agente: Se destinato ai modelli di chat oppure per definire applicazioni in cui si utilizzano gli LLM.

-
2. Supporto della Cronologia delle Chat: Se l'agente può essere utilizzato come chatbot, mantenendo una conversazione coerente nel tempo. Altrimenti, l'agente è più adatto a compiti singoli, senza la necessità di mantenere una cronologia delle conversazioni.
 3. Supporto di Strumenti con più Input: L'agente può gestire strumenti che richiedono più input contemporaneamente, migliorando la sua flessibilità.

Agent Type	Model Type	Chat History	Multi Input	When to Use
OpenAI Tools	Chat	✓	✓	Con modelli OpenAI recenti
OpenAI Functions	Chat	✓	✓	Modello OpenAI o un Open Source ottimizzato per la chiamata di funzioni
XML	LLM	✓		Modelli antropici o esperti in XML
Structured Chat	Chat	✓	✓	Necessario supportare più input
JSON Chat	Chat	✓		Modelli che forniscono risposte JSON
ReAct	LLM	✓		Modelli semplici
Self Ask With Search	LLM			Modelli semplici con l'utilizzo di un unico strumento di ricerca

Tabella 2.1: Quando utilizzare gli agenti. Fonte: langchain.com

La tabella 2.1 mostra quando utilizzare gli agenti, in base alla loro destinazione, al supporto per la cronologia delle chat e alla gestione di input multipli. [7]

Langchain è una piattaforma progettata per semplificare l'uso di modelli di linguaggio avanzati. Ci sono diversi motivi per preferire questa soluzione rispetto ad altre opzioni come LLaMa.cpp o GPT-4all:

1. Facilità d'uso: Langchain rende l'accesso ai modelli di linguaggio estremamente semplice. Con poche righe di codice, è possibile istanziare e utilizzare modelli come GPT-4 senza dover affrontare dettagli complessi.
2. Ampia gamma di modelli: Langchain offre una vasta selezione di modelli, inclusi modelli personalizzati come Falcon e Wizard. Questa varietà consente di scegliere il modello più adatto alle specifiche esigenze.
3. Nessuna GPU o connessione Internet richiesta: A differenza di altre soluzioni, Langchain non richiede una GPU o una connessione Internet per l'inferenza dei modelli.

-
- 4. Supporto per callback e streaming: Langchain offre supporto per callback e streaming, consentendo di ottenere output più dettagliati dai modelli. Ad esempio, è possibile ottenere previsioni token per token o personalizzare i parametri di generazione secondo le proprie esigenze.

In conclusione, Langchain rappresenta una solida scelta per lo sviluppo di applicazioni che utilizzano modelli linguistici avanzati, in modo rapido e senza eccessive complicazioni.

Capitolo 3

I Large Language Model

Un Large Language Model (LLM) è un modello linguistico che si distingue per la sua capacità di generare testo e svolgere altri compiti di elaborazione del linguaggio naturale. Gli LLM acquisiscono queste capacità apprendendo relazioni statistiche da documenti di testo durante un processo di addestramento auto-supervisionato e semi-supervisionato ad alta intensità computazionale [22]. Gli LLM possono essere utilizzati per la generazione di testi, una forma di IA generativa, che prendendo un testo in input ne prevedendo ripetutamente il token o la parola successiva in modo da formare una frase di senso compiuto [6]. L’architettura attualmente più diffusa per i modelli linguistici di intelligenza artificiale è il Transformer, che impiega il meccanismo dell’attenzione basata sull’apprendimento automatico per pesare selettivamente l’importanza dei diversi token nel fare la prossima previsione.

In sostanza, anche senza istruzioni esplicite, i computer possono estrarre informazioni dai dati, individuare correlazioni e ”imparare” il linguaggio. Man mano che apprende gli schemi in base a cui vengono messe in sequenza le parole, il modello può fare previsioni sulla struttura delle frasi basate su calcoli di probabilità. Alla fine di questo processo di addestramento, si ottiene un modello in grado di individuare relazioni complesse tra le parole e le frasi [13].

3.1 Applicazioni dei LLM

Dalle attività ripetitive all’esperienza dei clienti, i Large Language Models stanno aprendo nuove possibilità per migliorare l’efficienza, l’innovazione e la relazione con il pubblico.

- Scrittura di testi. Oltre a GPT-4o e ChatGPT, esistono altri modelli come Claude, Llama 2, Cohere Command e Jurassiccan che possono generare testi originali.

-
- Risposte basate sulle conoscenze. Spesso definita come elaborazione del linguaggio naturale ad alta intensità di conoscenza (KI-NLP), questa tecnica si riferisce ai LLM capaci di rispondere a domande specifiche utilizzando archivi digitali.
 - Classificazione del testo. Utilizzando il clustering, i LLM possono classificare testi con significati o sentimenti simili. Questo può essere utile per misurare il sentimento dei clienti, determinare la relazione tra i testi e condurre ricerche documentali.
 - Generazione di codice. I LLM sono capaci di generare codice da istruzioni in linguaggio naturale. Esempi includono Amazon CodeWhisperer e GitHub Copilot, che possono codificare in Python, JavaScript, Ruby e altri linguaggi di programmazione. Altre applicazioni includono la creazione di query SQL, la scrittura di comandi shell e la progettazione di siti web.
 - Generazione di testo. Simile alla generazione di codice, la generazione di testo permette ai LLM di completare frasi incomplete, scrivere documentazione di prodotto o scrivere brevi storie.

L'ampia gamma di applicazioni dei Large Language Models sta rivoluzionando numerosi settori, dalla generazione di testi alla risoluzione di compiti complessi di elaborazione del linguaggio naturale. Questi modelli aprono nuove prospettive per migliorare l'efficienza, stimolare l'innovazione e potenziare l'interazione con il pubblico. Dal supporto alla scrittura di testi originali alla generazione di codice e alla classificazione del testo, i LLM dimostrano un potenziale straordinario nell'affrontare una vasta gamma di sfide [2].

3.2 Training dei Modelli

Il training è il processo attraverso il quale si addestrano i Large Language Model, consentendo loro di comprendere e generare testo in modo efficace e corretto. Questo processo coinvolge l'uso di una vasta quantità di dati testuali (o di immagini) e l'applicazione di algoritmi di apprendimento, permettendo al modello di acquisire conoscenze linguistiche e semantiche. Il risultato è un sistema capace di produrre testi simili a quelli scritti dall'uomo, tradurre tra lingue diverse, rispondere a domande e svolgere una vasta gamma di compiti cognitivi.

Il termine "large" in LLM si riferisce al numero di parametri del modello. Questi parametri sono le variabili che il modello utilizza per fare previsioni. Il numero totale di parametri è generalmente definito dagli acronimi: 7b, 13b, 34b, ecc... significano rispettivamente "7 miliardi di parametri", "13 miliardi di parametri" e "34 miliardi di parametri". Più alto è il numero di parametri, maggiore è la complessità e la capacità

del modello di comprendere e generare testo, ma non è l'unica variabile che conta. Altri fattori, come la qualità dei dati di addestramento è un particolare importante.

Il training dei LLM si articola in diversi passaggi cruciali, tra cui la raccolta dei dati, la configurazione del modello, l'addestramento effettivo del modello e l'ottimizzazione finale. Ogni fase riveste un ruolo fondamentale nel garantire che il modello sia adeguatamente preparato, ottimizzato e pronto per svolgere compiti complessi e produrre risultati accurati.

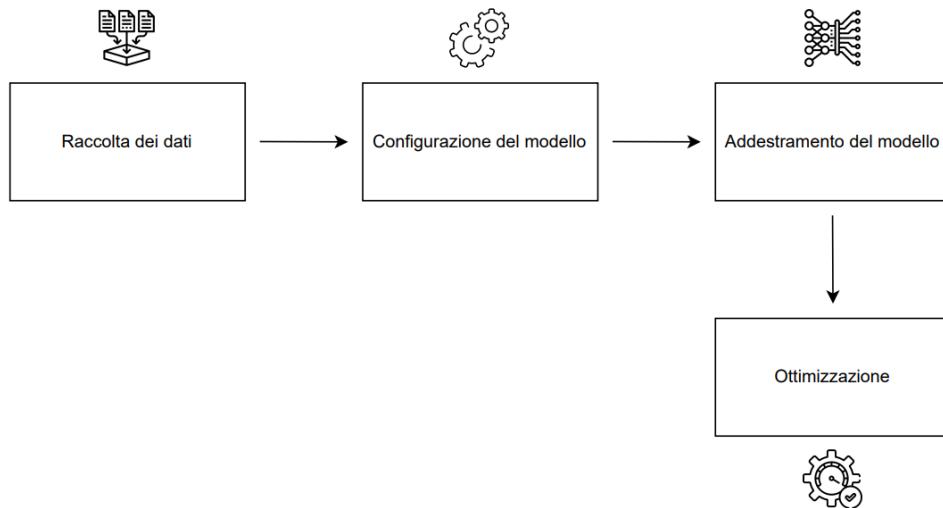


Figura 3.1: LLM-Training

1. **Raccolta dei dati:** Questo primo passo coinvolge la ricerca e la comprensione dell'insieme di dati di addestramento. I dati possono provenire da una vasta gamma di fonti come libri, articoli, contenuti web e set di dati open-access. È importante selezionare dati rappresentativi e diversificati per garantire una formazione completa e accurata del modello. Inoltre, i dati devono essere puliti e preparati per l'addestramento. Questo può includere operazioni come la conversione in minuscolo, la rimozione delle stop word e la tokenizzazione.
2. **Configurazione del modello:** I framework di Deep Learning Transformer sono comunemente utilizzati per le applicazioni di Elaborazione del Linguaggio Naturale. Quando si configura una rete neurale, è necessario definire determinati parametri. Questi possono includere il numero di strati all'interno dei blocchi del trasformatore, le attenzioni multiple e gli iperparametri. I ricercatori sperimentano tipicamente con i parametri per trovare una combinazione che produca prestazioni ottimali per il compito specifico.
3. **Addestramento del modello:** Una volta che i dati di testo puliti e preparati sono disponibili, possono essere utilizzati per addestrare il modello. Durante il processo

di addestramento, il modello riceve una serie di parole e cerca di prevedere la parola successiva nella sequenza data. Il modello modifica quindi le assegnazioni dei pesi in base alle sue previsioni successive di parole. Questo ciclo viene ripetuto molte volte, con il modello che viene esposto a milioni o addirittura miliardi di esempi, fino a quando non raggiunge prestazioni ottimali.

- Ottimizzazione: Una volta completato l'addestramento, il modello viene valutato utilizzando un set di dati di test per valutarne le prestazioni. A seconda dei risultati di questi test, possono essere apportati aggiustamenti di ottimizzazione al modello per migliorare le sue capacità predittive e di generalizzazione.

Questi passaggi sono fondamentali per garantire che il modello sia adeguatamente addestrato e possa svolgere compiti complessi nel modo più accurato e affidabile possibile. Il training dei LLM richiede tempo, risorse computazionali e una rigorosa metodologia per ottenere risultati soddisfacenti [2] [26].

3.3 Valutazione dei Modelli

Esistono diverse metriche standard per la valutazione degli LLM, tra cui perplessità, accuratezza, punteggio F1, punteggio ROUGE, punteggio BLEU, punteggio METEOR, metriche di risposta alle domande, metriche di analisi del sentimento, metriche di riconoscimento di entità denominate e incorporamenti di parole contestualizzate. Queste metriche aiutano a valutare le prestazioni LLM misurando vari aspetti del testo generato, come fluidità, coerenza, accuratezza e pertinenza.

La perplessità è una misura di quanto bene un modello linguistico predica un campione di testo. Viene calcolato come la probabilità inversa del set di test normalizzata per il numero di parole:

$$\text{Perplessità} = 2^{-\frac{\log P(w_1, w_2, \dots, w_n)}{n}}$$

Immaginiamo di avere un modello linguistico addestrato su un corpus di testo e di volerne valutare le prestazioni su un set di test. Il set di test è composto da 1000 parole e il modello linguistico assegna una probabilità di 0,001 a ciascuna parola. La perplessità del modello linguistico sul set di test è $2^{-\log(0,001 \times 1000)/1000} = 31,62$.

L'accuratezza è una misura della capacità di un modello linguistico di effettuare previsioni corrette. Viene calcolato come il numero di pronostici corretti diviso per il numero totale di pronostici:

$$\text{Precisione} = \frac{\text{numero di previsioni corrette}}{\text{numero totale di previsioni}}$$

Supponiamo di avere un modello linguistico addestrato a classificare immagini di cani e gatti. Testiamo il modello su una serie di 100 immagini, di cui 80 sono gatti e 20 sono cani. Il modello classifica correttamente 75 gatti e 15 cani. La precisione del modello è $\frac{75+15}{80+20} = 0,9$.

Il punteggio F1 è una misura dell'equilibrio di un modello linguistico tra precisione e richiamo. Si calcola come media armonica di precisione e richiamo:

$$\text{Punteggio F1} = \frac{2 \cdot \text{precisione} \cdot \text{richiamo}}{\text{precisione} + \text{richiamo}}$$

Supponiamo di avere un modello linguistico addestrato per identificare le e-mail di spam. Testiamo il modello su un set di 100 email, di cui 80 legittime e 20 spam. Il modello identifica correttamente 15 email di spam e identifica erroneamente 5 email legittime come spam. La precisione del modello è 0,75 e il richiamo del modello è 0,75. Il punteggio F1 del modello è $\frac{2 \times (0,75 \times 0,75)}{0,75 + 0,75} = 0,75$ [4].

3.3.1 Il Punteggio ROUGE

Il punteggio ROUGE valuta la capacità di un modello linguistico di generare testo simile ai riferimenti forniti. Esso analizza la similitudine tra il riassunto creato dal modello e i riferimenti utilizzando gli n-grammi, ossia sequenze di parole presenti sia nel riassunto prodotto che nei riferimenti. Questo punteggio calcola la proporzione di n-grammi presenti nel riassunto del modello rispetto ai riferimenti dati.

$$ROUGE = \sum \text{Recall of } n\text{-grams}$$

Il concetto di "Recall of n-grams" si riferisce al numero di n-grammi che compaiono sia nel riassunto generato dalla macchina che nei riepiloghi di riferimento, diviso per il numero totale di n-grammi nei riepiloghi di riferimento.

Il punteggio ROUGE varia da 0 a 1, dove valori più elevati indicano una maggiore qualità del riassunto. Un riassunto perfetto avrebbe un punteggio ROUGE di 1, mentre un riassunto completamente scorretto avrebbe un punteggio ROUGE di 0.

I punteggi ROUGE si suddividono in:

- ROUGE-N: Misura la sovrapposizione di n-grammi tra il testo candidato e il testo di riferimento. Calcola precisione, richiamo e punteggio F1 basandosi sulla sovrapposizione degli n-grammi. ROUGE-N è spesso utilizzato per valutare correttezza grammaticale e fluidità del testo generato.
- ROUGE-L: Misura la sottosequenza comune più lunga tra il testo candidato e il testo di riferimento. Calcola precisione, richiamo e punteggio F1 basandosi sulla

lunghezza della sottosequenza. ROUGE-L è spesso utilizzato per valutare somiglianza semantica e copertura del contenuto del testo generato, in quanto considera la sottosequenza comune indipendentemente dall'ordine delle parole.

- ROUGE-S: Misura la sovrapposizione skip-bigram (bigram con al massimo una parola intermedia) tra il testo candidato e il testo di riferimento. Calcola precisione, richiamo e punteggio F1 basandosi sulla sovrapposizione dei skip bigram. ROUGE-S è spesso utilizzato per valutare coerenza e coesione locale del testo generato, poiché cattura somiglianza semantica tra parole adiacenti.

3.3.2 Altre Metriche

Il punteggio BLEU è una misura della capacità di un modello linguistico di generare un testo fluido e coerente. Viene comunemente utilizzato per attività di generazione di testo come la traduzione automatica e la didascalia delle immagini.

Il punteggio METEOR misura la capacità di un modello linguistico di generare testo accurato e pertinente. Combina precisione e richiamo per valutare la qualità del testo generato [4].

Inoltre, i Large Language Model vengono sottoposti a valutazione su benchmark standardizzati, mostrati nella tabella 3.1 [15].

Benchmarks	Descrizione	URL di riferimento
GLUE Benchmark	Il benchmark GLUE (General Language Understanding Evaluation) fornisce un insieme standardizzato di diversi compiti di NLP per valutare l'efficacia dei modelli linguistici	https://gluebenchmark.com/
SuperGLUE Benchmark	Confronto tra diversi compiti più impegnativi utilizzando GLUE, con basi umane	https://super.gluebenchmark.com/
HellaSwag	Valuta la capacità di un LLM di completare una frase	https://rowanzellers.com/hellaswag/
TruthfulQA	Misura la veridicità delle risposte del modello	https://github.com/sylinrl/TruthfulQA
MMLU	MMLU (Massive Multitask Language Understanding) valuta la capacità dell'LLM di lavorare in multitasking.	https://github.com/hendrycks/test

Tabella 3.1: Benchmark valutazione prestazioni LLM. Fonte: medium.com

3.4 GPT4All vs LLaMa

Il panorama dei LLM è in continua evoluzione, caratterizzato dalla competizione tra diverse tecnologie di spicco. Due delle più rilevanti sono GPT4All e LLaMA.

GPT4All, sviluppato da Nomic, rappresenta un ecosistema completo di chatbot open source che fornisce un framework versatile per l'addestramento di modelli linguistici. Dall'altra parte, OpenLLaMA è un'iniziativa di OpenLM Research che mira a creare una versione senza restrizioni di LLaMa (una famiglia di modelli linguistici di grandi dimensioni autoregressivi, rilasciata da Meta AI [30]), adatta sia per applicazioni di ricerca che per scopi commerciali [9].

Local Large Memory Access (LLaMa) è un altro modello linguistico, che utilizza un numero inferiore di parametri rispetto ad alcuni modelli più grandi, come le versioni 13B, 30B e 65B, ma offre comunque prestazioni impressionanti [9].

LLaMA e GPT4All sono entrambi potenti modelli linguistici che sono stati addestrati tramite fine-tuning per fornire risultati di alta qualità per diversi compiti. LLaMA è considerato l'LLM più performante di Meta AI per ricercatori e per casi d'uso non commerciali. Si concentra sull'efficienza dei parametri rispetto ai grandi LLM commerciali, il che lo rende una scelta competitiva.

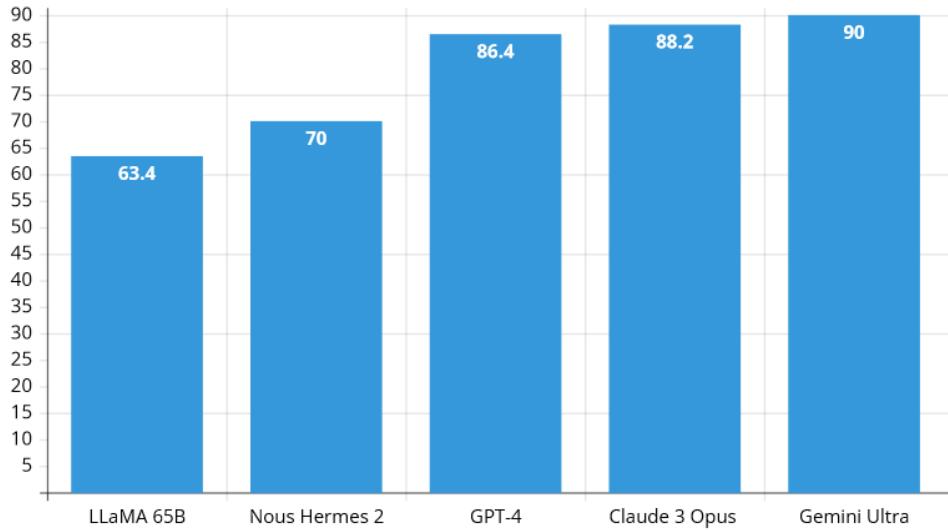


Figura 3.2: Punteggio MMUL. Fonte: paperswithcode.com [8]

Nella figura 3.2 sono rappresentati rispettivamente i principali modelli LLM locali e open-source come LLaMa 65B e Nous Hermes 2; e successivamente i migliori LLM proprietari come GPT-4, Cloude 3 Opus e Gemini Ultra.

Capitolo 4

Implementazione

Un possibile utilizzo dei Large Language Models riguarda la traduzione di dati sanitari rappresentati in formato JSON in linguaggio naturale comprensibile al personale sanitario. Ad esempio, GPT-4o può analizzare e interpretare le strutture dei dati JSON di FHIR, trasformando le informazioni in frasi leggibili e comprensibili. È importante valutare attentamente i rischi associati all'utilizzo di servizi terzi come OpenAI e adottare adeguate misure di mitigazione.

Il trasferimento di informazioni personali a servizi terzi come OpenAI comporta un rischio significativo di violazione della privacy. Anche se tali servizi implementano misure di sicurezza robuste, esiste sempre il rischio di violazioni dei dati o di accessi non autorizzati, che potrebbero compromettere la confidenzialità delle informazioni personali.

Inoltre, la trasmissione di dati sensibili a questi servizi terzi comporta la perdita di controllo diretto sulla gestione di tali informazioni. Non è possibile avere certezze sul modo in cui tali dati vengono manipolati e archiviati dall'azienda terza.

Un ulteriore rischio da considerare è la possibilità di fuga dei dati. Anche se un servizio come OpenAI può avere politiche rigorose sulla protezione dei dati, esiste sempre il rischio che i dati inviati possano essere oggetto di fuga o compromissione, sia attraverso vulnerabilità nel sistema del fornitore dei servizi, sia tramite attacchi informatici esterni.

Pertanto, si è scelto di procedere implementando una versione locale e open source dei large language model per mitigare tali rischi.

4.1 Analisi e Raccolta dei Requisiti

Nel processo di sviluppo di qualsiasi sistema o software, l'analisi e la raccolta dei requisiti sono fasi cruciali. Questa sezione delinea l'obiettivo e l'ambito di Medical-AI, specificando gli obiettivi e i criteri di successo. Si esamina inoltre il panorama attuale dei sistemi

disponibili per i professionisti medici e si fornisce una descrizione dettagliata del sistema proposto, inclusi gli attori coinvolti e i requisiti funzionali e non funzionali.

4.1.1 Scopo del sistema

Medical-AI è progettato per fornire assistenza medica rapida e accurata, consentendo ai medici di ottenere informazioni pertinenti e affidabili senza dover effettuare ricerche manuali attraverso database interni. Il sistema è in grado di accedere e ricercare dati quali:

- Elenco dei pazienti presenti nel sistema.
- Storico delle prescrizioni mediche per un determinato paziente.
- Dettagli sulle condizioni mediche di un paziente specifico.
- Elenco delle procedure chirurgiche pianificate per la settimana.
- Registrazione delle allergie dei pazienti.
- Lista delle terapie farmacologiche in corso per un paziente.
- Generazione di un rapporto dei risultati di laboratorio relativi a un paziente specifico, inclusa, se necessario, la visualizzazione di un grafico riassuntivo.
- Riepilogo delle diagnosi recenti dei pazienti.
- Registro delle visite ambulatoriali dei pazienti.
- Elenco delle risorse disponibili nel sistema (pazienti, medici, ecc.).
- Storico delle terapie fisiche prescritte per un paziente.
- Registro delle comunicazioni tra il personale medico e i pazienti.

4.1.2 Ambito del sistema

Il sistema supporta il lavoro medico consentendo interrogazioni del server FHIR tramite linguaggio naturale. Il personale medico può autenticarsi e utilizzare un'interfaccia user-friendly per accedere alle informazioni richieste. Le funzionalità includono:

- Consentire al personale medico di accedere alla piattaforma inserendo le proprie credenziali.

-
- Consentire al personale medico di modificare il server FHIR o la key di OpenAI a cui fare riferimento.
 - Calendario degli appuntamenti per le visite mediche.
 - Registrazione delle misurazioni cliniche effettuate sui pazienti.

4.1.3 Obiettivo e Criteri di Successo del Sistema

L'obiettivo principale di Medical-AI è ottimizzare la gestione dei dati sanitari. I criteri di successo sono definiti come segue:

- Efficacia: Il sistema deve eseguire accuratamente tutte le operazioni richieste.
- Interfaccia User-Friendly: L'interazione con il sistema deve essere semplice e immediata per garantire la soddisfazione degli utenti.
- Soddisfazione degli Stakeholders: Massimizzare la soddisfazione sia dei realizzatori che degli utenti finali.
- Interoperabilità: Il sistema deve integrarsi con altri sistemi e applicazioni.

Questo approccio mira a fornire un sistema completo e affidabile per migliorare l'efficienza e la qualità dell'assistenza medica.

4.1.4 Analisi dei sistemi correnti

Nella presente sezione, eseguiremo un'analisi dettagliata dei sistemi informatici attualmente utilizzati dal personale medico. Esamineremo le principali tecnologie impiegate, valutando i loro punti di forza, le aree di miglioramento e le sfide attuali.

Electronic Health Record (EHR)

Un Electronic Health Record (EHR) è una versione elettronica della storia clinica di un paziente, e può includere tutti i dati amministrativi e clinici chiave rilevanti per le cure di quella persona: demografia, note di progresso, problemi, farmaci, anamnesi patologica, immunizzazioni, dati di laboratorio e referti radiologici.

L'EHR può migliorare l'assistenza ai pazienti:

- Mettendo a disposizione le informazioni sanitarie, riducendo la duplicazione di test, diminuendo i ritardi nel trattamento e mantenendo i pazienti ben informati per prendere decisioni migliori.

-
- Riducendo gli errori medici e migliorando ulteriormente l'accuratezza e la chiarezza dei record medici.

Gli EHR rappresentano il prossimo passo nel progresso continuo dell'assistenza sanitaria che può rafforzare la relazione tra pazienti e clinici. I dati e la tempestività e disponibilità degli stessi consentiranno di prendere decisioni migliori e fornire cure migliori [27].

Picture Archiving and Communication System (PACS)

Il mondo della radiologia e dell'imaging medico ha subito una trasformazione epocale con l'avvento dei sistemi PACS.

Essi sfruttano algoritmi sofisticati per acquisire e digitalizzare le immagini provenienti da una vasta gamma di dispositivi di imaging, trasformando segnali fisici in dati digitali ad alta definizione. Questo processo di digitalizzazione non solo riduce il rischio di deterioramento delle immagini, ma consente anche una rapida trasmissione e archiviazione su supporti digitali.

Una volta acquisite, le immagini vengono immagazzinate in repository digitali sicuri, dove la tecnologia informatica entra in gioco per organizzare e indicizzare le immagini in modo efficiente. Grazie a sofisticati algoritmi di archiviazione e ricerca, i professionisti sanitari possono accedere istantaneamente a immagini specifiche, recuperando dati storici o eseguendo ricerche avanzate basate su criteri specifici.

Uno dei vantaggi più significativi dell'adozione dei sistemi PACS è la sua capacità di facilitare la condivisione delle informazioni. Attraverso reti sicure e protocolli di comunicazione avanzati, le immagini mediche possono essere condivise istantaneamente tra professionisti sanitari, consentendo consulenze rapide e decisioni collaborative. Questo è particolarmente prezioso in contesti complessi come la telemedicina e la collaborazione tra ospedali e centri di ricerca [31].

Health Information Exchange (HIE)

I sistemi di Health Information Exchange (HIE) rivoluzionano la condivisione delle informazioni sanitarie, migliorando l'accesso, la sicurezza e l'efficienza sanitaria.

Al cuore degli HIE risiedono sofisticati sistemi informatici che consentono la raccolta, l'archiviazione e la trasmissione sicura dei dati sanitari dei pazienti. Questi sistemi utilizzano standard di interoperabilità, come HL7 e FHIR, per garantire che le informazioni possano essere scambiate senza intoppi tra diverse piattaforme e sistemi sanitari, eliminando così le barriere alla condivisione delle informazioni.

Complessi algoritmi crittografici e protocolli di autenticazione garantiscono che solo gli utenti autorizzati possano accedere ai dati sensibili dei pazienti, proteggendo così la loro privacy e riservatezza. Inoltre, i sistemi di monitoraggio continuo e la rilevazione delle intrusioni garantiscono una difesa proattiva contro minacce esterne e interne alla sicurezza dei dati.

Un altro aspetto cruciale dell'informatica negli HIE è la gestione delle interoperabilità dei dati. Gli HIE devono essere in grado di integrarsi con una vasta gamma di sistemi informatici utilizzati da ospedali, cliniche, laboratori e altre organizzazioni sanitarie. Questo richiede un'architettura flessibile e scalabile, in grado di adattarsi ai diversi standard e protocolli utilizzati da queste piattaforme.

Inoltre, l'analisi dei dati è diventata un elemento fondamentale degli HIE, consentendo agli operatori sanitari di estrarre conoscenze significative dai vasti quantitativi di dati clinici raccolti. Tecniche avanzate di data mining e intelligenza artificiale vengono impiegate per identificare pattern, tendenze e correlazioni nei dati, fornendo preziose informazioni per il miglioramento delle cure e la prevenzione delle malattie [29].

Fast Healthcare Interoperability Resources (FHIR)

L'adozione diffusa di Fast Healthcare Interoperability Resources (FHIR) ha rivoluzionato il modo in cui i dati sanitari vengono gestiti e condivisi nell'ambito sanitario. Grazie alla sua struttura modulare e alle API basate su REST, FHIR si è affermato come uno standard fondamentale nell'interoperabilità dei sistemi sanitari, consentendo un accesso più rapido, sicuro e efficiente alle informazioni cliniche.

Al centro dell'informatica nei sistemi basati su FHIR vi è l'architettura delle API RESTful, che permette agli sviluppatori di creare applicazioni sanitarie altamente interoperabili. Queste API semplificano la comunicazione tra sistemi diversi, consentendo lo scambio fluido di dati tra piattaforme eterogenee. Ciò significa che le informazioni cliniche possono essere facilmente accessibili e aggiornate in tempo reale, migliorando così la coordinazione delle cure e la qualità dell'assistenza fornita ai pazienti.

Inoltre, l'adozione di FHIR ha aperto la strada a nuove opportunità nell'ambito dell'analisi dei dati sanitari. Gli algoritmi avanzati possono esplorare grandi quantità di dati clinici strutturati secondo lo standard FHIR, identificando pattern, tendenze e correlazioni che possono guidare decisioni cliniche informate e migliorare i risultati dei pazienti. Questa capacità di analisi avanzata consente agli operatori sanitari di trarre vantaggio da un ricco pool di dati per migliorare la prevenzione, la diagnosi e il trattamento delle malattie.

Un altro aspetto cruciale dell'informatica nei sistemi FHIR è la sicurezza e la privacy dei dati. Poiché le informazioni sanitarie sono estremamente sensibili, è fondamentale

garantire che siano protette da accessi non autorizzati e da eventuali violazioni della privacy. FHIR include meccanismi di sicurezza avanzati, come la crittografia dei dati e l'autenticazione a più fattori, che assicurano che solo gli utenti autorizzati possano accedere alle informazioni cliniche dei pazienti [28].

4.1.5 Il Sistema Proposto

Come evidenziato, i sistemi attuali generano dati strutturati che possono essere complessi da interpretare rapidamente per il personale medico. Perciò, l'adozione di un Large Language Model (LLM) per tradurre le occorrenze FHIR in linguaggio naturale potrebbe offrire notevoli vantaggi nel settore sanitario. È proprio questo lo scopo di Medical-AI:

- Semplificazione delle Informazioni Tecniche: Le occorrenze FHIR contengono dati clinici strutturati e altamente tecnici che possono risultare complessi. Tradurre queste informazioni in linguaggio naturale può rendere più accessibili e comprensibili i dati clinici, consentendo una migliore comunicazione tra i medici.
- Supporto alla Decisione Clinica: Un LLM addestrato su dati sanitari potrebbe essere in grado di fornire analisi contestuali e interpretazioni utili delle informazioni cliniche contenute nelle occorrenze del server. Questo potrebbe supportare i medici nel processo decisionale clinico, fornendo suggerimenti e raccomandazioni basati sull'evidenza per la diagnosi e il trattamento dei pazienti.
- Risparmio di Tempo per i Medici: La traduzione automatica delle occorrenze FHIR in linguaggio naturale potrebbe ridurre il tempo necessario per interpretare e spiegare i dati clinici ai pazienti. Ciò consentirebbe ai medici di concentrarsi maggiormente sull'assistenza diretta ai pazienti anziché sulla gestione dei compiti amministrativi, migliorando l'efficienza complessiva delle cure.

Requisiti Funzionali

- Traduzione dei Dati Clinici: Il sistema deve essere in grado di tradurre le occorrenze JSON FHIR in linguaggio naturale, fornendo una descrizione comprensibile dei dati clinici.
- Supporto per Diverse Occorrenze FHIR: Il sistema deve supportare una varietà di tipi di occorrenze FHIR, inclusi i dati demografici del paziente, le anamnesi mediche, i risultati dei test di laboratorio, le prescrizioni e altro ancora.
- Interfaccia Utente Intuitiva: Il sistema dovrebbe fornire un'interfaccia utente intuitiva e facile da usare per gli utenti finali, consentendo loro di visualizzare i risultati della traduzione in modo chiaro e organizzato.

-
- Integrazione con Sistemi Esistenti: Il sistema dovrebbe essere in grado di integrarsi con altri sistemi sanitari esistenti, consentendo l'importazione e l'esportazione agevole dei dati clinici tradotti.

Requisiti Non Funzionali

- Precisione: Il sistema deve garantire un'alta precisione nella traduzione delle occorrenze, minimizzando gli errori e le inesattezze nella descrizione dei dati clinici.
- Velocità di Risposta: Il sistema dovrebbe fornire traduzioni rapide delle occorrenze FHIR, garantendo tempi di risposta rapidi per gli utenti.
- Sicurezza dei Dati: Il sistema deve garantire la sicurezza e la riservatezza dei dati clinici durante il processo di traduzione e la gestione delle informazioni.
- Affidabilità: Il sistema dovrebbe essere affidabile e disponibile in modo continuo per gli utenti, minimizzando i tempi di inattività e garantendo la disponibilità dei servizi.
- Scalabilità: Il sistema deve essere in grado di gestire carichi di lavoro variabili e aumentare la capacità in modo dinamico per adattarsi alla crescente domanda degli utenti.
- Usabilità: L'interfaccia utente del sistema dovrebbe essere intuitiva e facile da usare, riducendo al minimo la necessità di formazione aggiuntiva per gli utenti.
- Conformità Normativa: Il sistema dovrebbe essere conforme alle normative e ai regolamenti sulla privacy dei dati, come l'Health Insurance Portability and Accountability Act (HIPAA) negli Stati Uniti o il General Data Protection Regulation (GDPR) in Europa.

4.2 Fase di progettazione di Medical-AI

In questa fase, si procede con una dettagliata descrizione delle attività coinvolte nella realizzazione di Medical-AI. Si inizia con la specifica degli obiettivi del sistema, seguita dall'analisi dell'architettura proposta. Infine, si delinea il meccanismo di gestione e coordinamento del flusso di esecuzione e del comportamento del sistema software nel suo complesso.

4.2.1 Gli obiettivi del sistema

Gli obiettivi del sistema delineano le direttive principali per lo sviluppo di un assistente virtuale innovativo che integrerà le potenzialità dell'intelligenza artificiale con le esigenze del settore medico. In questa sezione, esamineremo attentamente gli obiettivi fondamentali che guidano la progettazione e l'implementazione di Medical-AI.

- **Interfaccia Intuitiva:** Creare un'interfaccia utente intuitiva che permetta agli utenti di interagire con il sistema attraverso il linguaggio naturale.
- **Traduzione di Linguaggio Naturale:** Utilizzare l'intelligenza artificiale fornita da OpenAI per tradurre in modo accurato e coerente il linguaggio naturale delle richieste degli utenti in richieste per il server FHIR.
- **Integrazione con FHIR:** Garantire un'efficace integrazione con il server FHIR, permettendo al sistema di inviare e ricevere dati in modo sicuro e affidabile.
- **Elaborazione delle Risposte:** Interpretare le risposte dal server FHIR sotto forma di dati JSON e strutturarle in frasi di senso compiuto utilizzando un LLM, al fine di renderle comprensibili e utili per gli utenti.
- **Precisione e Affidabilità:** Assicurare la precisione e l'affidabilità del sistema nell'interpretare le richieste degli utenti, nell'elaborare le risposte dal server FHIR e nella generazione di frasi di senso compiuto.

4.2.2 Architettura del Sistema

L'architettura del sistema è basata sul modello a tre strati Model-View-Controller (MVC), con l'aggiunta di componenti aggiuntivi come template engine e gestione delle risorse statiche.

Il Model gestisce i dati dell'applicazione, interagendo con il database. È implementato attraverso un insieme di classi Python che forniscono metodi per l'accesso e la manipolazione dei dati con l'utilizzo della libreria TinyDB.

La view è responsabile della presentazione dei dati all'utente e delle interazioni utente. Le view sono state implementate come funzioni Python che gestiscono le richieste HTTP e generano risposte. Le view sono state utilizzate per generare pagine HTML in base alle richieste dell'utente.

Il controller coordina il flusso dell'applicazione, gestendo le richieste dell'utente e chiamando i metodi appropriati nei model per ottenere o manipolare i dati. Il controller è stato implementato come moduli separati, per una più corretta organizzazione.

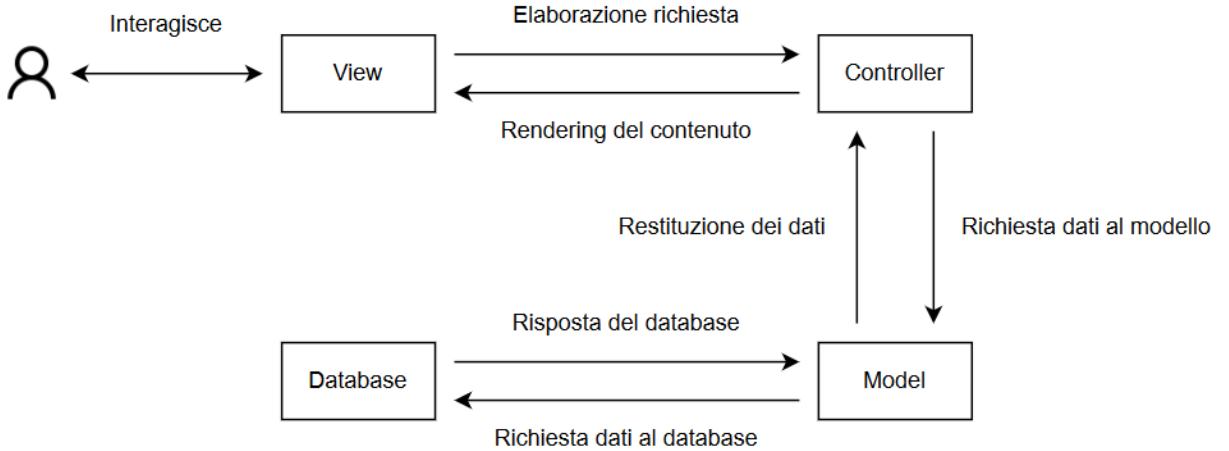


Figura 4.1: Il Modello MVC.

È stato utilizzato anche il template engine Jinja2 per la generazione dinamica delle pagine HTML. Questi template consentono di separare la logica di presentazione dalla logica di business dell'applicazione.

4.3 Lo sviluppo di Medical-AI

Lo sviluppo di Medical-AI ha visto all'adozione di diverse tecnologie. Questa sezione offre una panoramica dell'architettura utilizzata nel sistema proposto.

Per lo sviluppo del front-end sono stati utilizzati:

- **HTML**: Utilizzato per la definizione della struttura delle pagine web, HTML fornisce gli elementi base per la creazione dell'interfaccia utente.
- **CSS**: Applicato per lo stile degli elementi definiti con HTML, CSS consente di definire l'aspetto visivo e l'impaginazione delle pagine web.
- **JavaScript**: Aggiunge interattività e dinamicità alle pagine web, consentendo la gestione degli eventi utente e la manipolazione del DOM.

Per il back-end dell'applicazione, sono stati impiegati il linguaggio di programmazione Python insieme al framework Flask e alla libreria TinyDB.

Python è stato scelto come linguaggio principale per lo sviluppo del back-end grazie alla sua sintassi chiara e concisa, alla vasta gamma di librerie disponibili e alla sua flessibilità. Flask è un framework web per Python che fornisce strumenti per la creazione rapida di applicazioni web. Grazie alla sua semplicità e leggerezza, Flask è stato utilizzato per la gestione delle richieste HTTP e la definizione delle route dell'applicazione. TinyDB è

una semplice libreria di database per Python che è stata scelta per la gestione dei dati nel back-end dell'applicazione. Essa offre una soluzione leggera e facile da utilizzare per memorizzare e recuperare i dati.

Più nello specifico, per lo sviluppo in Pyhton sono stati utilizzati:

- Un server FHIR che implementa il protocollo di interoperabilità Fast Healthcare Interoperability Resources (FHIR).
- L'API di OpenAI, utilizzata per accedere ai modelli GPT (Generative Pre-trained Transformer), che sono strumenti potenti per compiti di generazione di testo. Questo ci consente di integrare funzionalità avanzate di linguaggio naturale nell'applicazione Medical-AI. Grazie a questa integrazione, l'applicazione può generare query e risposte accurate, migliorando così la sua capacità di supportare processi decisionali clinici e fornire assistenza sanitaria più efficace.
- L'API di LangChain, un framework utilizzato per l'utilizzo di modelli LLM LLaMa. Grazie a LangChain, è stata facilitata l'integrazione e l'interazione con i modelli LLM all'interno dell'architettura del sistema.

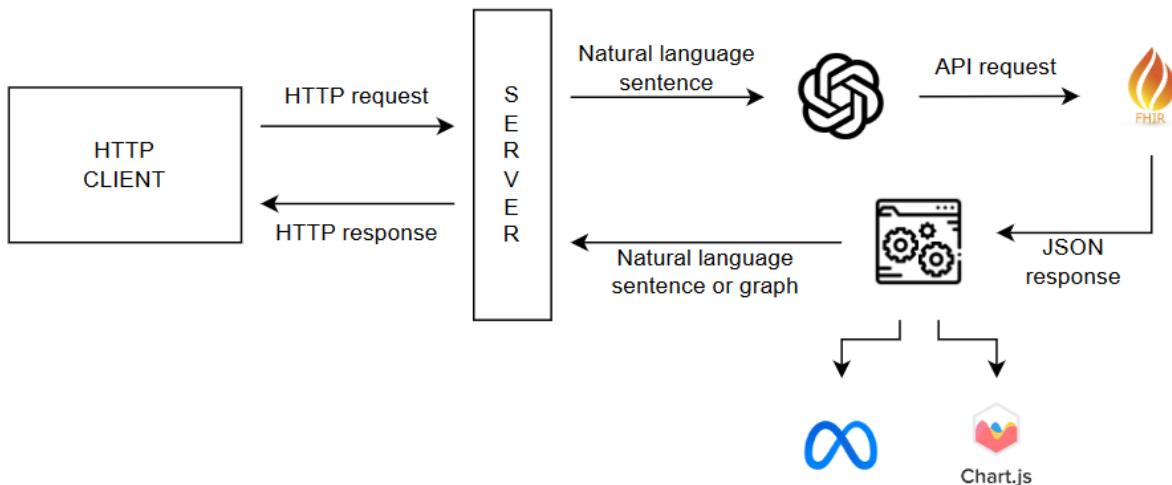


Figura 4.2: Architettura

4.3.1 API di OpenAI

Le API di OpenAI, utilizzando il modello GPT-4o, hanno permesso lo sviluppo di una funzionalità per interrogare il server FHIR. Questa funzionalità consente la generazione automatica di query al server FHIR per ottenere le informazioni necessarie. Una volta completata la fase iniziale di interrogazione tramite le API di OpenAI, il sistema passa

all'esecuzione in locale. Ciò significa che il software continua autonomamente a interrogare il server e gestire le risposte ricevute, tutto all'interno dell'ambiente locale senza ulteriori dipendenze esterne.

4.3.2 Utilizzo di LangChain

L'applicazione del framework LangChain ha aiutato nello sviluppo delle diverse funzionalità previste da Medical-AI. L'interrogazione del server FHIR è una funzionalità sviluppata grazie all'utilizzo del framework LangChain, il quale ha la capacità di orchestrare una serie di istruzioni al fine di ottenere il risultato desiderato.

LangChain & LlamaCpp

Attraverso LangChain, è stata utilizzata la libreria LlamaCpp-Python per l'inferenza locale dei modelli linguistici. Utilizzando la funzione `LlamaCpp`, è stato istanziato un oggetto chiamato `llm`, attraverso il quale è possibile effettuare l'inferenza. Questa funzione ci consente di configurare vari parametri, tra cui:

- Temperatura: Questo parametro consente di regolare la "creatività" del modello. Quando la temperatura è inferiore a 1, la differenza relativa tra i punteggi dei token aumenta, rendendo le scelte più deterministiche. Al contrario, una temperatura superiore a 1 riduce la differenza relativa tra i punteggi dei token, rendendo le scelte meno deterministiche.
- Top K: Questo parametro imposta un limite sul numero massimo di token che il modello può selezionare. Ad esempio, se si imposta `top_k = 5`, il modello può scegliere solo tra i primi 5 candidati e ignorare gli altri. Questo metodo è considerato piuttosto superficiale poiché tronca le scelte in base a un numero fisso di opzioni.
- Top P: Questo metodo coinvolge la somma dei punteggi dei token fino a raggiungere una percentuale target. Ad esempio, se abbiamo punteggi del 25%, 25%, 12.5%, 12.5%, e impostiamo `top_p = 0.50`, verranno considerati solo i primi due candidati, poiché la loro somma raggiunge il 50% del totale. Questo approccio è utile per fornire un grado di flessibilità nella selezione dei token senza dover specificare un numero fisso.
- Context Length: I large language model sono addestrati con una lunghezza di context fissa, che rappresenta il numero massimo di token considerati in una singola istanza di input. Ad esempio, il modello base Llama 2 è stato addestrato per supportare fino a 4096 token di contesto alla volta. Questo determina quanto il modello può

”guardare” indietro durante la generazione del testo, influenzando la comprensione del contesto e la coerenza nella produzione del linguaggio.

- Max Tokens: Il parametro indica il numero massimo di token nell’input, sommato all’output generato dal modello.

Per l’applicazione Medical-AI sono stati impostati questi parametri:

```
1     llm=LlamaCpp(  
2         model_path="../models/mistral-7b-openorca.Q6_K.gguf",  
3         temperature=0,  
4         max_tokens=500,  
5         n_ctx=2048,  
6         top_p=1,  
7     )
```

Figura 4.3: Impostazioni per l’Inferenza Locale.

Modelli LLaMA

Per lo sviluppo del sistema, sono stati considerati vari modelli open source basati sull’architettura llama. Nella tabella 4.1 sono elencati i modelli testati e confrontati con modelli proprietari, accompagnati dal rispettivo punteggio MMLU ottenuto [8].

L’analisi comparativa ha permesso di valutare le prestazioni dei modelli open source in termini di velocità di elaborazione e capacità di generare output di alta qualità. Questa analisi è stata cruciale per identificare il modello più adatto alle esigenze di implementazione.

Model	MMUL
Llama 2 7B	45.3
Mistral 7B	60.1
Falcon 7B	28.0
GPT-4*	86.4
GPT-3.5 Turbo*	70.0
Gemini Ultra*	90.0

Tabella 4.1: Benchmark valutazione prestazioni LLM locali.

* I modelli indicati sono proprietari, e non sono stati considerati.

Anche se i modelli proprietari possono vantare una potenza considerevole, presentano una serie di problemi. Esistono diverse ragioni per prediligere l'utilizzo di modelli open source e locali per la conversione dei dati in linguaggio naturale.

I modelli open source sono accessibili senza restrizioni di licenza e possono essere modificati e migliorati dalle community, e quindi è possibile personalizzarli per gestire in modo più efficace i dati sanitari, che possono essere complessi e ricchi di termini tecnici. Inoltre, è possibile addestrare il modello con dati locali per migliorare le prestazioni nelle specifiche aree o lingue.

Inoltre l'utilizzo di modelli locali per l'elaborazione dei dati sanitari comporta diversi vantaggi in termini di sicurezza e privacy dei dati. Quando i dati vengono elaborati su server remoti, c'è sempre il rischio che possano essere soggetti a intercettazioni o violazioni durante il trasferimento attraverso la rete. Questo è particolarmente critico quando si tratta di informazioni sensibili come i dati sanitari, che possono includere informazioni personali altamente riservate.

Con l'adozione di modelli locali, l'elaborazione dei dati avviene direttamente sul dispositivo o all'interno dell'ambiente di rete dell'organizzazione, eliminando la necessità di trasmettere i dati a server esterni. Questo significa che i dati rimangono sotto il controllo diretto dell'organizzazione e non vengono esposti a potenziali rischi durante il trasferimento attraverso Internet.

Inoltre, l'elaborazione locale può essere accompagnata da misure di sicurezza aggiuntive, come crittografia dei dati e controlli di accesso rigorosi, per garantire che solo le persone autorizzate possano accedere alle informazioni sensibili. Questo livello aggiuntivo di sicurezza contribuisce a proteggere i dati sanitari da accessi non autorizzati o violazioni della sicurezza.

Un altro vantaggio dell'elaborazione locale è la possibilità di rispettare le normative sulla privacy e la protezione dei dati, che possono variare da regione a regione. Ad esempio, in alcuni paesi, ci sono restrizioni rigide sul trasferimento di dati personali al di fuori dei confini nazionali. Utilizzando modelli locali, le organizzazioni possono garantire la conformità normativa e ridurre il rischio di sanzioni legali associate alla violazione delle leggi sulla privacy dei dati.

Mistral 7B

Mistral 7B, il più recente modello di Mistral AI, rappresenta un notevole avanzamento nell'ambito dei modelli linguistici, caratterizzato da un'architettura potente e prestazioni superiori rispetto ai suoi predecessori. Con 7.3 miliardi di parametri, Mistral 7B eccelle in una varietà di compiti, dalla comprensione del linguaggio naturale al ragionamento su codice, offrendo una soluzione all'avanguardia per le sfide dell'intelligenza artificiale.

Il modello Mistral 7B si distingue per la sua maggiore precisione e coerenza nelle risposte rispetto agli altri modelli testati. Questi ultimi hanno presentato notevoli problemi di allucinazioni, ovvero la generazione di informazioni false o fuorvianti, e risposte non pertinenti.

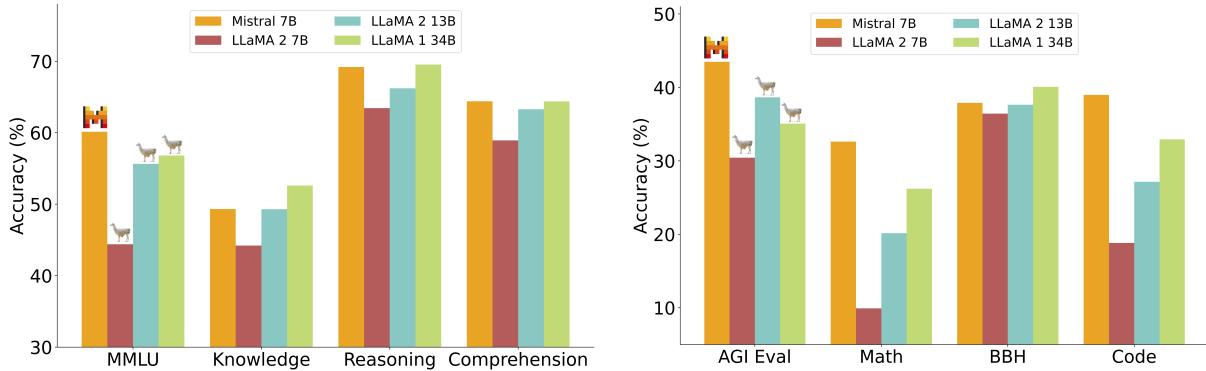


Figura 4.4: Prestazioni di Mistral 7B e di diversi modelli di Llama. Fonte: mistral.ai [21]

Una delle caratteristiche distintive di Mistral 7B è la sua capacità di superare Llama 2 13B su tutti i benchmark, oltre a offrire prestazioni paragonabili a Llama 1 34B su diversi compiti. Questo modello si distingue anche per l'implementazione di tecnologie avanzate, come la Grouped Query attention (GQA) per un'elaborazione più rapida delle richieste e la Sliding Window Attention (SWA) per gestire sequenze più lunghe con costi computazionali ridotti.

Un aspetto significativo di Mistral 7B è la sua licenza Apache 2.0, che consente agli sviluppatori di utilizzarlo senza restrizioni, sia in ambienti locali che su piattaforme cloud come AWS, GCP e Azure. Inoltre, il modello è facilmente adattabile a qualsiasi compito tramite il fine-tuning del modello.

Nel dettaglio delle prestazioni, Mistral 7B ha dimostrato superiorità su una vasta gamma di benchmark, coprendo aree come il ragionamento comune, la conoscenza del mondo, la comprensione della lettura, la matematica e la stesura di codice. Grazie al meccanismo di Sliding Window Attention, il modello ottimizza l'efficienza computazionale, consentendo una maggiore velocità di elaborazione senza compromettere la qualità dell'output [21].

In particolare nel progetto di Medical-AI, è stato utilizzato un modello derivato dall'utilizzo del dataset di OpenOrca [19] per ottimizzare il modello originale Mistral 7B. Le valutazioni della classifica Hugging Face posizionano questo modello al primo posto tra tutti i modelli inferiori a 30B al momento del rilascio, superando tutte le altre varianti 7B

e 13B. Il modello ha ottenuto un punteggio medio MMUL di 65.84, rispetto al punteggio di 62.4 del modello originale, confermando un miglioramento significativo delle prestazioni.

ChatPromptTemplate

Nel contesto di Medical-AI, i prompt sono strumenti fondamentali per guidare il modello nella creazione di output coerenti e pertinenti. Essi consentono di fornire istruzioni o input specifici per indirizzare la risposta del modello in base al contesto e agli obiettivi desiderati. Di fatto, il cuore del lavoro di Medical-AI risiede proprio nell'utilizzo efficace dei prompt.

Per la generazione dei prompt è stata utilizzata la classe `ChatPromptTemplate` tramite il framework LangChain. Il prompt quindi diventa una lista di messaggi che formano la chat. Ogni messaggio di chat è associato a un contenuto e a un parametro aggiuntivo chiamato `role`.

I ruoli possibili sono:

- 1.Utente: un messaggio "user" viene inviato al modello, che poi recupera il vettore di incorporamento per quel messaggio e trova le migliori corrispondenze nel modello di base.
- 2.Assistente: un messaggio "assistant" è un testo aggiuntivo che viene combinato con il messaggio utente. Il messaggio rappresenta la risposta dell'AI al messaggio utente.
- 3.Sistema: un messaggio "system" è un messaggio che lo sviluppatore ha scritto specificando al bot come interpretare la conversazione.

4.3.3 Prompt Engineering

Il prompt engineering è il processo di guida delle risposte dell'AI generativa verso risultati desiderati attraverso l'uso di prompt specifici. Anche se l'AI generativa cerca di imitare il comportamento umano, ha bisogno di istruzioni dettagliate per produrre risultati rilevanti e di alta qualità. Questa pratica richiede la scelta accurata di formati, frasi, parole e simboli appropriati per guidare l'AI nell'interazione con gli utenti in modo significativo.

Le funzionalità offerte da Medical-AI sono state realizzate mediante l'adozione di un prompt unico e specifico per ciascun obiettivo da raggiungere, garantendo così una risposta mirata e coerente con le necessità dell'utente.

Generazione delle Richieste per il Server FHIR

Per l'implementazione dell'assistente virtuale si è utilizzato OpenAI con il modello generativo GPT-4o. Il modello ha dimostrato grande padronanza nel generare query specifiche da effettuare al server FHIR, con un basso tasso di errore.

```
1     messages=[  
2         {"role":"system", "content":f"You are a URL generator for  
3             ↳ my FHIR server{session['fhir_url']}. Provide only the  
4             ↳ full URL to execute the request for the doctor  
5             ↳ considering that he has all the permissions to do so. For  
6             ↳ example, if you are asked for the list of patients,  
7             ↳ provide '/baseR4/Patient'."},  
8         {"role":"user", "content": user_message}  
9     ]
```

Figura 4.5: Prompt per la generazione di query FHIR.

Nel prompt illustrato nella figura 4.5, viene utilizzato un messaggio `system` per indicare le azioni richieste al modello: generare degli URL per effettuare richieste al server FHIR. Dato che le risposte possono essere non deterministiche, useremo successivamente le espressioni regolari (REGEX) per estrarre solo l'URL fornito. Inoltre, è incluso un esempio nel prompt per guidare il modello a rispondere in modo simile. Il messaggio `user` consiste nelle richieste dell'utente, che possono essere tradotte in richieste HTTP oppure possono essere utilizzate per ottenere risposte generiche dal modello di OpenAI.

Generazione di Frasi in Linguaggio Naturale

A causa delle preoccupazioni legate alla sicurezza nell'utilizzo di servizi esterni per la generazione delle risposte, si è deciso di optare per l'inferenza locale utilizzando i modelli LLaMa, in particolare il modello Mistral-7B. Questa scelta consente di mantenere i dati sensibili all'interno del proprio ambiente, riducendo i rischi associati alla trasmissione di informazioni su reti esterne.

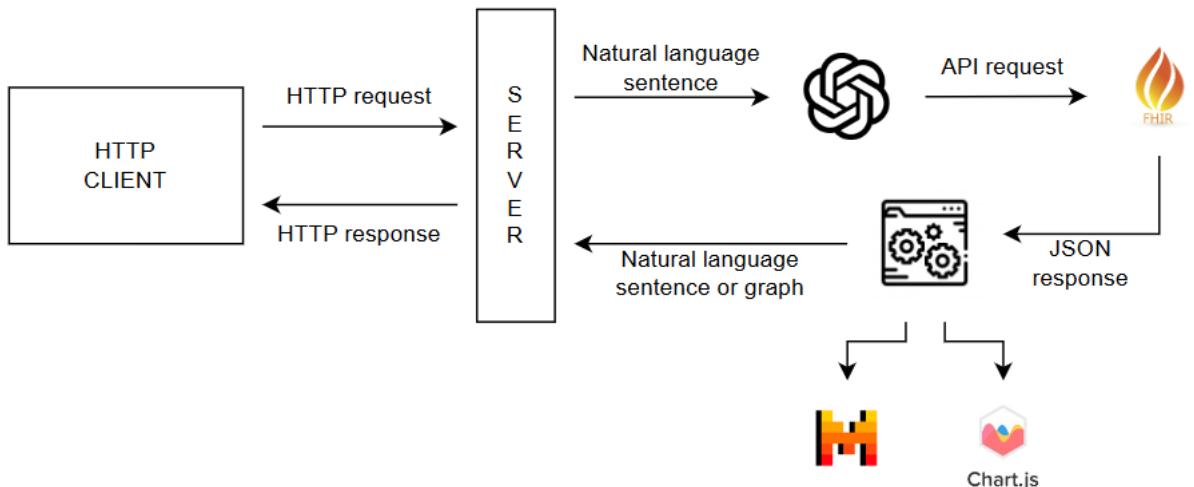


Figura 4.6: Architettura di Medical-AI

La generazione dell'URL per le richieste API al server è gestita dalla libreria di OpenAI, utilizzando il modello GPT-4o. Questo processo include la creazione e formattazione delle richieste API necessarie per interagire con il server remoto. Una volta che la richiesta è stata inviata e la risposta è stata ricevuta dal server, i dati ricevuti vengono elaborati localmente. Tale elaborazione viene effettuata tramite il modello Mistral-7B, eseguito all'interno del framework LangChain, che consente di sfruttare le potenzialità del modello in modo sicuro e controllato. Questo approccio garantisce un equilibrio tra l'utilizzo di risorse esterne per le richieste API e il trattamento sicuro dei dati sensibili attraverso l'inferenza locale.

```

1   prompt=ChatPromptTemplate.from_messages(
2     [
3       ("system", "You are a translator of JSON code into
4         ↳ natural language sentences, eliminating unnecessary
5         ↳ attributes and provide only the generate sentence."),
6       ("user", "{patient_data}"),
7       ("assistant", "The identifier of this patient within the
8         ↳ hospital system is '12345'. His name is John Adam
9         ↳ Doe, he is male and was born on 25 December 1974. He
10        ↳ can be contacted at home on 555-555-5555."),
11       ("user", "{json}"),
12     ]
13   )

```

Figura 4.7: Prompt per l'esecuzione locale di Mistral-7B.

La figura 4.7 mostra il prompt utilizzato per l'inferenza. Come specificato, il primo messaggio inviato è di tipo `system`, nel quale si chiarisce il compito che il modello dovrà svolgere: tradurre le frasi dalla trascrizione JSON al linguaggio naturale. Il primo messaggio `human`, insieme alla risposta corrispondente dell'`assistant`, fornisce al modello un esempio di come dovrebbe rispondere. Infine, il secondo messaggio `human` fornisce il JSON effettivo che il modello dovrà tradurre.

L'elaborazione locale, quando utilizza un prompt di esempio, produce 0.54 token al secondo, mentre senza prompt di esempio raggiunge 1.33 token al secondo. Utilizzando una GPU T4, l'implementazione locale senza prompt di esempio ha generato 2.45 token al secondo, mentre con prompt di esempio ha raggiunto 1.75 token al secondo. Tutte le implementazioni sono basate su una quantizzazione Q6 nel formato GGUF, ottimizzato per il caricamento rapido dei modelli, rendendolo altamente efficiente per scopi di inferenza.

4.4 Esempio di Esecuzione di Medical-AI

Una volta effettuato l'accesso alla web-app, il personale medico può avviare il bot e formulare una richiesta in linguaggio naturale. Questa richiesta viene inviata al modello GPT-4o di OpenAI tramite le API proprietarie utilizzando il prompt mostrato in figura 4.5. OpenAI elaborerà la richiesta e potrà fornire due tipi di risposta: un percorso FHIR per effettuare richieste al server oppure una risposta diretta alla domanda, che sarà mostrata all'utente. Se viene restituito un URL FHIR, verrà eseguita una richiesta GET ad

esso; i risultati JSON ottenuti saranno elaborati localmente nel sistema di Medical-AI. Se ritenuto opportuno, il sistema creerà un grafico tramite Chart.js oppure scomporrà le voci del JSON e le invierà al LLM locale istanziando un oggetto *ChatPromptTemplate* con il relativo prompt mostrato in figura 4.7 per tradurle in linguaggio naturale.

4.5 Manuale Utente

Questa sezione è dedicata a illustrare l'utilizzo di Medical-AI e facilitare la sua comprensione da parte dell'utente. Qui vengono forniti dettagli sui passaggi necessari per sfruttare appieno le funzionalità offerte da Medical-AI, garantendo così un'esperienza ottimale durante l'utilizzo del software.

4.5.1 Accesso a Medical-AI

Quando un utente accede al sito, si troverà di fronte all'interfaccia di accesso per Medical-AI, come illustrato nella figura 4.8. L'accesso è consentito solo agli addetti previamente autorizzati dall'amministratore.

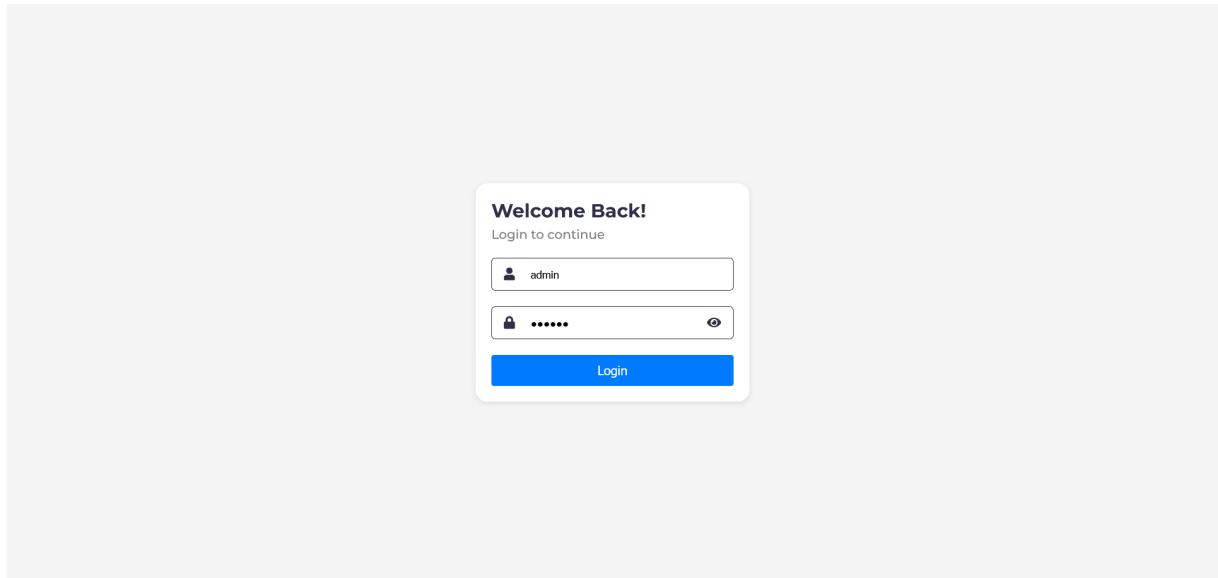


Figura 4.8: Schermata accesso a Medical-AI

Se le credenziali inserite (email e/o password) non sono corrette, verrà visualizzato un messaggio di errore, come illustrato nella figura 4.9.

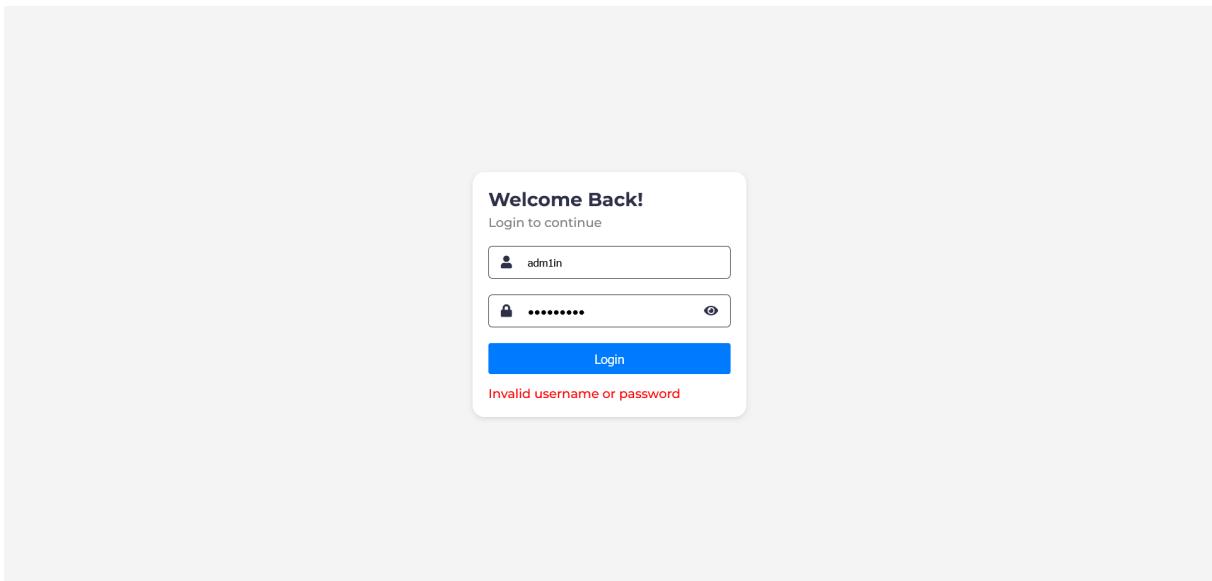


Figura 4.9: Errore nell'accesso a Medical-AI

Nel caso in cui le credenziali inserite siano corrette, l'utente sarà reindirizzato alla pagina principale 4.10.

A screenshot of the Medical-AI dashboard. At the top, there's a navigation bar with links for Home, Appointments, Patients, Observation, Account, and a search icon. Below the navigation, a welcome message "Welcome, admin!" is displayed next to the current date and time "2024-06-29 (Saturday) 12:00:07".

The left side of the dashboard features a "Patients" section with a list of five patients:

- Patient ID: 593222
Name: Jack O'Neil
Birth Date: 1967-10-31
Gender: male
- Patient ID: 593224
Name: Tiffany Borer
Birth Date: Not defined
Gender: female
- Patient ID: 593225
Name: Cody Nielsen
Birth Date: 2000-08-9
Gender: male
- Patient ID: 593223
Name: Luke Taylor
Birth Date: 1998-01-27
Gender: male
- Patient ID: 593253
Name: Benjamin Cook
Birth Date: 1981-12-15
Gender: male

The right side features an "Observations" section with a list of five observations:

- Observation ID: 1596340
Last Updated: 2024-06-27 09:28
Reference ID: Patient/1596139
Observation Type: Respiratory Rate
- Observation ID: 1596339
Last Updated: 2020-10-13 11:09
Reference ID: Patient/1596139
Observation Type: Body Height
- Observation ID: 1596341
Last Updated: 2020-10-13 11:09
Reference ID: Patient/1596139
Observation Type: Body Weight
- Observation ID: 1596342
Last Updated: 2020-10-13 11:09
Reference ID: Patient/1596139
Observation Type: Body Mass Index
- Observation ID: 1596343
Last Updated: 2020-10-13 11:09
Reference ID: Patient/1596139
Observation Type: Body mass index (BMI) [Percentile] Per age and gender

A blue speech bubble icon is located in the bottom right corner of the dashboard area.

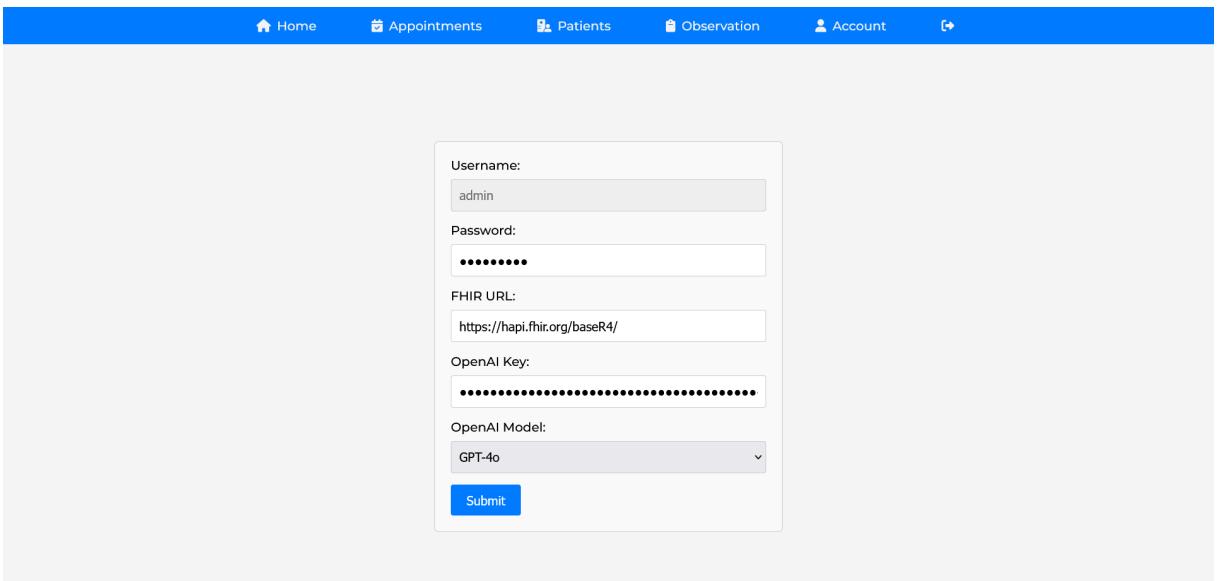
Figura 4.10: Dashboard di Medical-AI

Qui, troverà visualizzate le informazioni più recenti riguardanti i pazienti inseriti e le ultime analisi caricate nel sistema.

Le informazioni vengono visualizzate attraverso un parsing dei dati presenti nel server FHIR utilizzando il formato JSON.

4.5.2 Profilo Utente

Con l'apposito pulsante è possibile accedere al proprio account personale. Qui è possibile effettuare modifiche alla password di accesso, all'URL del server FHIR a cui l'assistente virtuale farà riferimento e al token di accesso a OpenAI. Inoltre sarà possibile modificare il modello con cui verrà effettuata la generazione degli URL scegliendo tra GPT-3.5-Turbo e GPT-4o 4.11.



The screenshot shows a web-based application interface with a blue header bar containing navigation links: Home, Appointments, Patients, Observation, Account, and a user icon. Below the header is a large white input form. The form fields include:

- Username: admin
- Password: (redacted)
- FHIR URL: https://hapi.fhir.org/baseR4/
- OpenAI Key: (redacted)
- OpenAI Model: GPT-4o (selected from a dropdown menu)

A blue "Submit" button is located at the bottom right of the form.

Figura 4.11: Profilo Personale in Medical-AI

4.5.3 Accesso all'Assistente Virtuale

L'accesso all'assistente virtuale è possibile tramite il pulsante posizionato nell'angolo in basso a destra. Cliccandolo, si aprirà la sezione contenente la chat, l'area di immissione dei messaggi e i pulsanti per chiudere oppure inviare il messaggio 4.12.

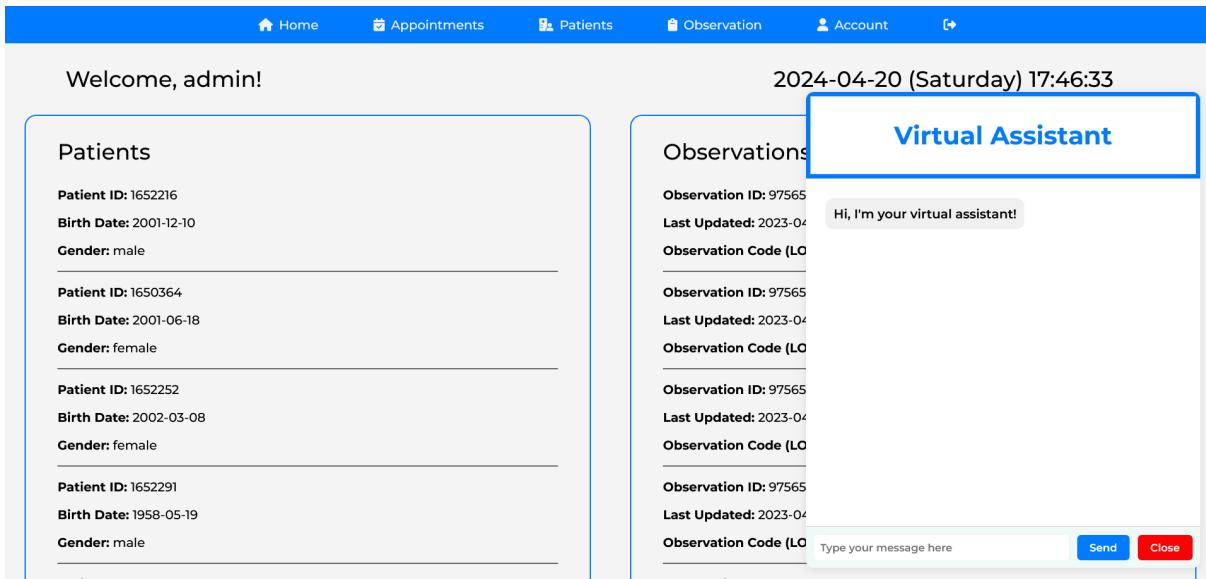


Figura 4.12: Assistente Virtuale di Medical-AI

L’assistente virtuale può rispondere a domande generali e, in particolare, è in grado di gestire richieste FHIR 4.13.

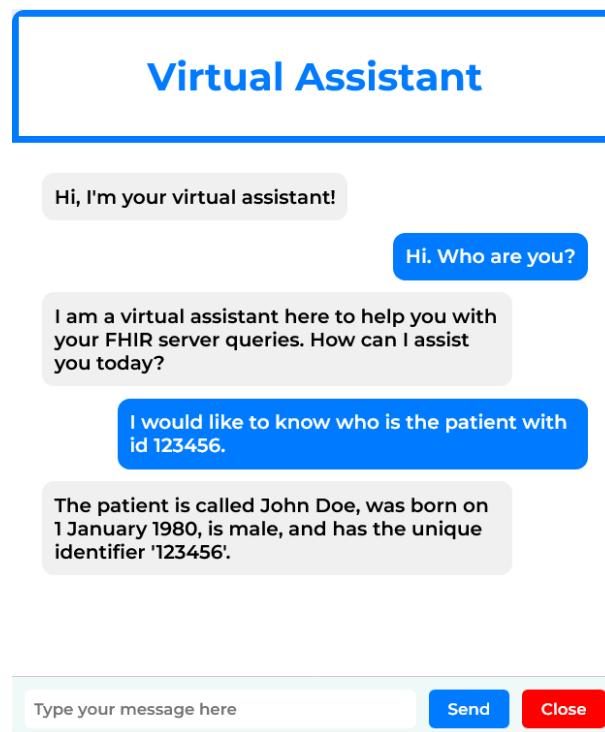


Figura 4.13: Esempio di Risposta.

La risposta mostrata nella figura 4.13 è stata elaborata a partire dal frammento JSON indicato 4.14. Questo JSON è strutturato per descrivere le informazioni di un paziente.

Nel nostro caso specifico, il paziente si chiama "Laoxian Cushing" ed è nato il 04-05-1999; nel sistema ospedaliero, è identificato con l'ID "592522".

```
1  {
2      "resourceType": "Patient",
3      "id": "592522",
4      "meta": {"lastUpdated": "2023-08-24T15:30:54.833+00:00"},
5      "identifier": [
6          {
7              "type": {
8                  "coding": [
9                      {
10                         "system": "http://hl7.org/fhir/v2/0203",
11                         "code": "MR"
12                     }
13                 ]
14             },
15             "value": "e090c850-60a9-46bf-b539-0e609ff45c1d"
16         }
17     ],
18     "name": [
19         {
20             "family": "Cushing",
21             "given": ["Laoxian"]
22         }
23     ],
24     "birthDate": "1999-05-04"
25 }
```

Figura 4.14: Esempio paziente.

In aggiunta, qualora fosse necessario, il sistema può visualizzare grafici relativi alle misurazioni cliniche eseguite sul paziente 4.15.

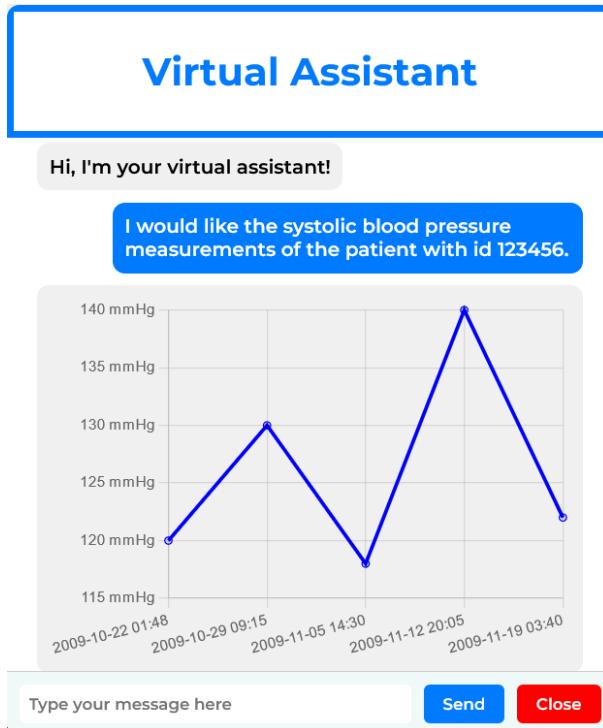


Figura 4.15: Esempio di Generazione di un Grafico

Altri esempi di risposte sono illustrati nella Figura 4.16. Questi includono rispettivamente:

- 1.Misurazione del peso del paziente con ID 12345.
- 2.Dettagli dell'ultima visita medica registrata per il paziente identificato come "xyz".
- 3.Elenco delle allergie riscontrate nel paziente con numero identificativo 93456.
- 4.Pazienti memorizzati nel sistema con il nome "Mario Rossi", inclusi eventuali dettagli associati alle loro cartelle cliniche.

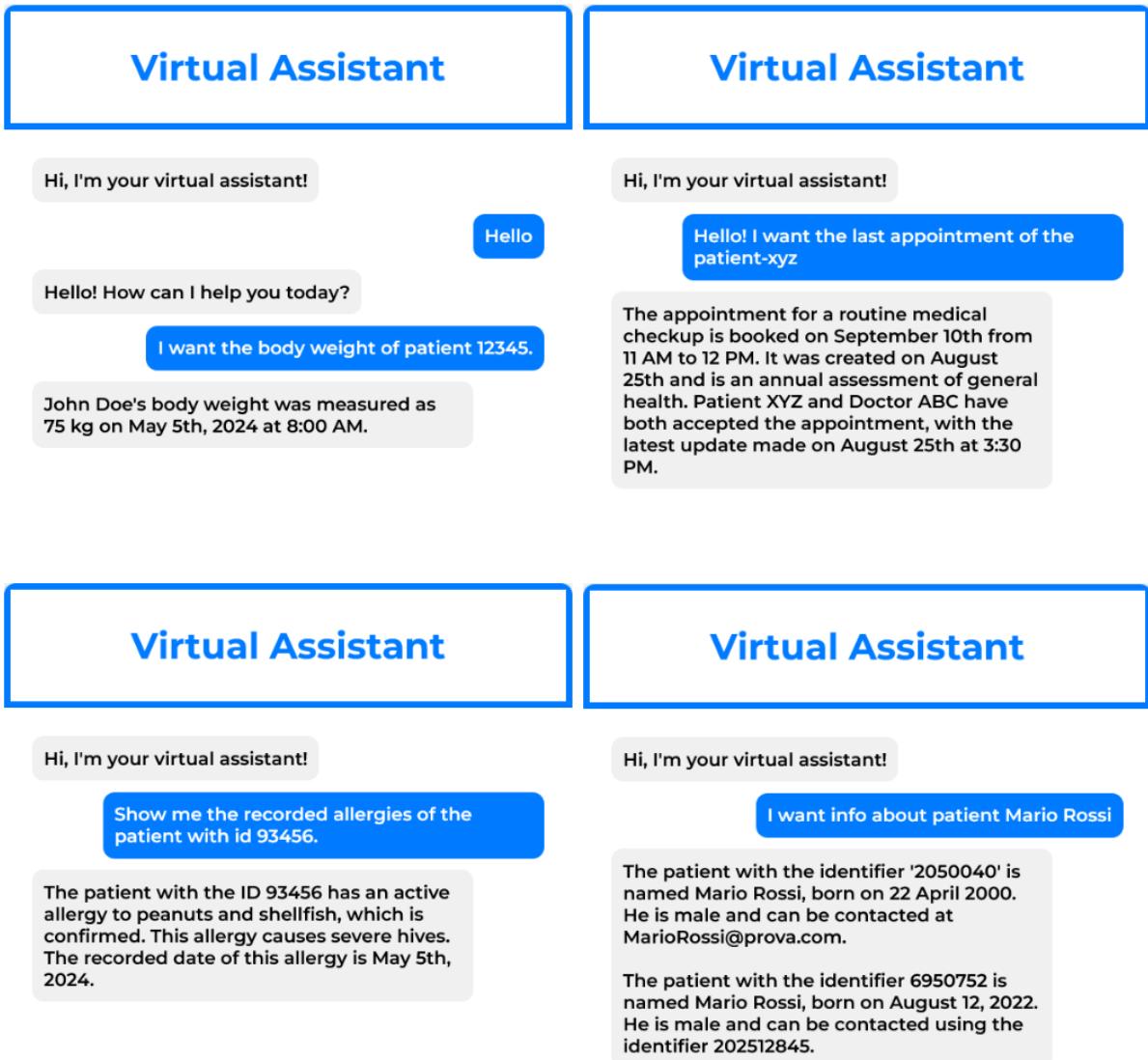


Figura 4.16: Altri Esempi.

4.6 Valutazione dei Risultati

Tramite l'utilizzo della libreria Pydantic sono state valutate le risposte ottenute dal modello. Pydantic è uno strumento potente e flessibile per la validazione e la gestione dei dati in Python, consentendo di definire modelli di dati rigorosi che rispecchiano le aspettative sui formati e sui tipi di dati generati da un LLM. Questa capacità di Pydantic di garantire la conformità dei dati ai modelli predefiniti è essenziale per assicurare che le risposte del modello siano accurate e coerenti.

L'uso di Pydantic in questo contesto non solo garantisce che i dati dei pazienti siano completi e accurati, ma permette anche una facile serializzazione e deserializzazione dei stessi, facilitando l'integrazione con altri sistemi e applicazioni.

È stata definita una funzione `calculate_equality` che calcola la percentuale di attributi corrispondenti tra due oggetti di tipo `generated` e `reference`. La funzione è definita come segue:

```
1 def calculate_equality(generated, reference):
2     if type(generated) != type(reference):
3         raise ValueError("Instances must be of the same type")
4
5     dict1 = generated.dict()
6     dict2 = reference.dict()
7
8     total_attributes = len(dict2)
9     if total_attributes == 0:
10        return 100.0
11
12     matching_attributes = 0
13     for key in dict1.keys():
14         if key in dict2 and dict1[key] == dict2[key]:
15             matching_attributes += 1
16
17     percentage = (matching_attributes / total_attributes) * 100.0
18     return percentage
```

Figura 4.17: Valutazione Percentuale di Somiglianza.

Dove:

- `generated` e `reference` sono due oggetti dello stesso tipo.
- `total_attributes` rappresenta il numero totale di attributi nel dizionario `reference`.
- `matching_attributes` è il numero di attributi che hanno gli stessi valori sia in `generated` che in `reference`.

La funzione inizia controllando se i due oggetti sono dello stesso tipo. Se non lo sono, solleva un'eccezione di tipo `ValueError`. Successivamente, converte gli oggetti in dizionari per facilitare il confronto degli attributi. Se il numero totale di attributi è

zero nel dizionario `reference`, la funzione restituisce direttamente 100.0, indicando una corrispondenza perfetta quando non ci sono attributi da confrontare.

Nel caso in cui ci siano attributi da confrontare, la funzione itera attraverso le chiavi del dizionario `generated`. Per ogni chiave, verifica se la chiave corrisponde anche nel dizionario `reference` e se i valori corrispondenti sono uguali. Conta quante corrispondenze perfette ci sono e calcola quindi la percentuale di attributi che corrispondono.

La percentuale di corrispondenza è calcolata come il rapporto tra il numero di attributi corrispondenti e il numero totale di attributi nel dizionario `reference`, moltiplicato per 100.0.s

Nella figura 4.18 è illustrata una classe Pydantic specifica per la valutazione dei pazienti. Questa classe `Patient` rappresenta un modello di dati utilizzato per descrivere informazioni di base su un paziente, conforme allo standard FHIR. Include vari campi opzionali come `id` per l'identificazione univoca del paziente, `active` per indicare lo stato attivo del paziente, `name` per i nomi del paziente, `gender` per il sesso, `birthDate` per la data di nascita, `deceasedBoolean` per indicare se il paziente è deceduto, `deceasedDateTime` per la data/ora del decesso, `address` per gli indirizzi del paziente, `telecom` per i contatti come telefono ed email, e `maritalStatus` per lo stato civile.

```

1   classPatient(BaseModel):
2       resourceType:str=Field("Patient", description="Tipo di risorsa,
3           ↵  fisso a 'Patient' per paziente FHIR")
4       id: Optional[str]=Field(None, description="Identificatore univoco
5           ↵  del paziente")
6       active: Optional[bool]=Field(None, description="Stato attivo del
7           ↵  paziente (True o False)")
8       name: Optional[List[str]]=Field(None, description="Lista dei nomi
9           ↵  del paziente")
10      gender: Optional[str]=Field(None, description="Sesso del paziente
11          ↵  (male, female, other, unknown)")
12      birthDate: Optional[str]=Field(None, description="Data di nascita
13          ↵  del paziente nel formato 'YYYY-MM-DD'")
14      deceasedBoolean: Optional[bool]=Field(None, description="Indica se
15          ↵  il paziente è deceduto (True o False)")
16      deceasedDateTime: Optional[str]=Field(None, description="Data/ora
17          ↵  del decesso del paziente nel formato ISO 8601")
18      address: Optional[List[Dict[str,str]]]=Field(None,
19          ↵  description="Array degli indirizzi del paziente")
20      telecom: Optional[List[str]]=Field(None, description="Array dei
21          ↵  contatti del paziente (telefono, email, ecc.)")
22      maritalStatus: Optional[str]=Field(None, description="Stato civile
23          ↵  del paziente (single, married, divorced, widowed, etc.)")

```

Figura 4.18: Classe Patient.

Nell'esempio 4.19 è presentato un paziente di nome John Doe, con identificativo "12345", di sesso maschile, nato il 15 maggio 1980, attualmente attivo, residente all'indirizzo "123 Main St, Atlanta, USA". Il paziente non è deceduto ed è stato segnalato come "married". I suoi contatti includono il numero di telefono "123-456-7890" e l'indirizzo email "john.doe@example.us".

```

generated = Patient(
    id="12345",
    name=["John", "Doe"],
    gender="male",
    birthDate="1980-05-15",
    address=["123 Main St", "Atlanta", "USA"],
    telecom=["123-456-7890", "john.doe@example.us"],
    maritalStatus="married"
)

reference = Patient(
    id="12345",
    active=True,
    name=["John", "Doe"],
    gender="male",
    birthDate="1980-05-15",
    deceasedBoolean=False,
    address=["123 Main St", "Atlanta", "USA"],
    telecom=["123-456-7890", "john.doe@example.us"],
    maritalStatus="married"
)

```

Percentuale di uguaglianza: 81.82%

Figura 4.19: Punteggio Traduzione - Patient John Doe

La figura 4.20 mostra una classe Pydantic per la creazione di un'istanza di "Allergy-Intolerance". In questa classe, **id** rappresenta l'identificatore univoco dell'intolleranza, mentre **patient** indica l'ID del paziente a cui è associata tale intolleranza. Il campo **substance** specifica la sostanza alla quale il paziente è allergico. Lo stato dell'intolleranza è indicato dal campo **status**, che nel caso specifico è attivo. La data in cui è stata registrata l'intolleranza è rappresentata dal campo **recordedDate**. Infine, **note** fornisce una descrizione della storia della reazione allergica del paziente.

```
1 class AllergyIntolerance(BaseModel):
2     resourceType:str=Field("AllergyIntolerance", description="Tipo di
3         ↵   risorsa, fisso a 'AllergyIntolerance' per allergie/intolleranze")
4     id: Optional[str]=Field(None, description="Identificatore univoco
5         ↵   della risorsa allergia/intolleranza")
6     patient: Optional[str]=Field(None, description="Riferimento al
7         ↵   paziente associato")
8     substance: Optional[str]=Field(None, description="Sostanza
9         ↵   scatenante l'allergia")
10    status: Optional[str]=Field(None, description="Stato della
11        ↵   condizione (active, inactive, resolved)")
12    recordedDate: Optional[str]=Field(None, description="Data di
13        ↵   registrazione della condizione nel formato ISO 8601")
14    note: Optional[str]=Field(None, description="Note aggiuntive sulla
15        ↵   condizione")
```

Figura 4.20: Classe AllergyIntolerance.

In Figura 4.21 è mostrato il codice Python per la creazione di un’istanza di ‘AllergyIntolerance’ per il paziente con ID ”12345”. La condizione è attiva e la sostanza segnalata è la penicillina. La data di registrazione è il 1 giugno 2023 e c’è una nota che indica che il paziente ha una storia di reazione allergica alla penicillina.

```
generated = AllergyIntolerance(
    id="allergy001",
    patient="12345",
    substance="Penicillin",
    status="active",
    recordedDate="2023-06-01",
)

reference = AllergyIntolerance(
    id="allergy001",
    patient="12345",
    substance="Penicillin",
    status="active",
    recordedDate="2023-06-01",
    note="Patient has a history of allergic reaction to
          penicillin."
)

Percentuale di uguaglianza: 85.71%
```

Figura 4.21: Punteggio Traduzione - Allergy Intolerance

Nella figura 4.22 è mostrata la classe Appointment. Essa è utilizzata per rappresentare un appuntamento nel contesto di un sistema FHIR. Questi attributi consentono di definire dettagli come il tipo di risorsa (**resourceType**), l’identificatore univoco dell’appuntamento (**id**), lo stato dell’appuntamento (**status**), una descrizione (**description**), le date di inizio (**start**) e fine (**end**) dell’appuntamento nel formato ISO 8601, l’elenco dei partecipanti (**participant**) che può includere medici e pazienti, e infine il luogo dell’appuntamento (**location**).

```

1 class Appointment(BaseModel):
2     resourceType:str=Field("Appointment", description="Tipo di
3         ↵    risorsa, fisso a 'Appointment' per gli appuntamenti")
4     id: Optional[str]=Field(None, description="Identificatore univoco
5         ↵    dell'appuntamento")
6     status: Optional[str]=Field(None, description="Stato
7         ↵    dell'appuntamento (pending, booked, arrived, fulfilled,
8         ↵    cancelled)")
9     description: Optional[str]=Field(None, description="Descrizione
10        ↵    dell'appuntamento")
11     start: Optional[str]=Field(None, description="Data e ora di inizio
12        ↵    dell'appuntamento nel formato ISO 8601")
13     end: Optional[str]=Field(None, description="Data e ora di fine
14        ↵    dell'appuntamento nel formato ISO 8601")
15     participant: Optional[List[Dict[str,str]]]=Field(None,
16         description="Elenco dei partecipanti all'appuntamento (es.
17         ↵    medico, paziente)")
18     location: Optional[str]=Field(None, description="Luogo
19         ↵    dell'appuntamento")

```

Figura 4.22: Classe Appointment.

Nella figura 4.23 è ripostato un esempio di un'appuntamento con ID "appointment001", prenotato per una visita di follow-up il 15 luglio 2023 dalle 10:00 alle 11:00 presso l'ospedale A. I partecipanti includono il dottor Smith nel ruolo di medico e John Doe come paziente.

```

generated = Appointment(
    id="appointment001",
    status="booked",
    description="Follow-up visit",
    start="2023-07-15T10:00:00",
    end="2023-07-15T11:00:00",
    participant=["Dr. Smith", "John Doe"],
    location="Hospital A"
)

reference = Appointment(
    id="appointment001",
    status="booked",
    description="Follow-up visit",
    start="2023-07-15T10:00:00",
    end="2023-07-15T11:00:00",
    participant=["Dr. Smith", "John Doe"],
    location="Hospital A"
)

```

Percentuale di uguaglianza: 100.0%

Figura 4.23: Punteggio Traduzione - Appointment

Nella figura 4.24 è illustrata una classe Pydantic specifica per la valutazione delle risposte relative alle visite mediche. La classe `Observation` include attributi essenziali: `resourceType` specifica il tipo fisso '`Observation`'. `id`: Identifica univocamente l'osservazione. `subject`: Riferisce al soggetto dell'osservazione (es. ID del paziente). `status`: Indica lo stato dell'osservazione (final, amended, corrected). `code`: Codice che descrive il tipo di misurazione. `valueQuantity`: Valore numerico dell'osservazione. `valueString`: Valore testuale dell'osservazione. `effectiveDateTime`: Data e ora in cui è stata effettuata l'osservazione (ISO 8601). `performer`: Entità che ha eseguito l'osservazione (es. nome del medico).

```

1   classObservation(BaseModel):
2       resourceType:str=Field("Observation", description="Tipo di
3           ↵  risorsa, fisso a 'Observation' per le osservazioni")
4       id: Optional[str]=Field(None, description="Identificatore univoco
5           ↵  dell'osservazione")
6       subject: Optional[str]=Field(None, description="Riferimento al
7           ↵  soggetto dell'osservazione (es. ID del paziente)")
8       status: Optional[str]=Field(None, description="Stato
9           ↵  dell'osservazione (final, amended, corrected, etc.)")
10      code: Optional[str]=Field(None, description="Codice
11          ↵  dell'osservazione (es. tipo di misurazione)")
12      valueQuantity: Optional[str]=Field(None, description="Valore
13          ↵  quantitativo dell'osservazione")
14      valueString: Optional[str]=Field(None, description="Valore testuale
15          ↵  dell'osservazione")
16      effectiveDateTime: Optional[str]=Field(None, description="Data/ora
17          ↵  effettiva dell'osservazione nel formato ISO 8601")
18      performer: Optional[str]=Field(None, description="Entità che ha
19          ↵  eseguito l'osservazione (es. nome del medico)")

20

```

Figura 4.24: Classe Observation.

L'esempio in figura 4.25 mostra un'osservazione con ID "observation001", relativa al soggetto identificato con il numero "12345". Lo stato dell'osservazione è definito come "final", e si riferisce alla misurazione della pressione sanguigna con il codice "blood-pressure". Il valore quantitativo registrato è di "120/80 mmHg". Questa osservazione è stata effettuata il 1° giugno 2023 alle 08:00 da parte del dottor Johnson.

```
generated = Observation(
    id="observation001",
    subject="12345",
    status="final",
    code="blood-pressure",
    valueQuantity="120/80 mmHg",
    effectiveDateTime="2023-06-01T08:00:00",
)

reference = Observation(
    id="observation001",
    subject="12345",
    status="final",
    code="blood-pressure",
    valueQuantity="120/80 mmHg",
    effectiveDateTime="2023-06-01T08:00:00",
    performer="Dr. Johnson"
)
```

Percentuale di uguaglianza: 88.89%

Figura 4.25: Punteggio Traduzione - Observation

I risultati ottenuti evidenziano l'efficacia del modello nel processo di analisi e sintesi delle informazioni contenute nell'originale JSON, trasformandole in frasi comprensibili e coerenti.

Capitolo 5

Conclusioni

Lo studio si è concentrato sull'esplorare le potenziali applicazioni del framework Lang-Chain nel campo delle soluzioni sanitarie basate sui Large Language Model. Il risultato è Medical-AI, una piattaforma web progettata per sfruttare appieno le capacità dei Large Language Model al fine di fornire supporto al personale medico nella ricerca dei dati nei server FHIR.

Durante lo sviluppo di Medical-AI, sono stati integrati sia il modello di linguaggio GPT-4o di OpenAI tramite le loro API dedicate, sia il modello Mistral-7B con architettura LLaMa utilizzando le API fornite da LangChain.

La sfida principale è stata selezionare il modello più adatto al caso d'uso, in modo che l'applicazione fosse accessibile sulla maggior parte dei computer consumer, senza richiedere hardware sofisticato o l'invio dei dati sanitari privati a servizi terzi.

Inoltre, un altro aspetto fondamentale è stato formulare prompt efficaci per ottenere risposte pertinenti e coerenti, riducendo al minimo il rischio di errore. Ogni funzionalità implementata ha richiesto la progettazione di prompt specifici, mirati alle esigenze particolari di quella funzione. Questo approccio ha consentito di esprimere i compiti in linguaggio naturale, rappresentando un nuovo paradigma di "programmazione" che ha svolto un ruolo chiave nello sviluppo di Medical-AI.

I punteggi ottenuti nella fase di valutazione del sistema riflettono la notevole efficacia del modello nell'analizzare e riassumere una vasta gamma di informazioni, dimostrando una capacità non solo di trattare dati dei pazienti e appuntamenti medici, ma anche di comprendere dettagli specifici come le condizioni mediche e i relativi sintomi. Questi risultati suggeriscono che il modello sia in grado di elaborare e sintetizzare dati complessi in modo chiaro e conciso.

La buona corrispondenza tra i riassunti generati e quelli di riferimento indica una capacità del modello di catturare accuratamente i dettagli importanti e di esprimere tali informazioni in modo comprensibile. Questo è un aspetto cruciale nel contesto sanitario,

dove la precisione e la chiarezza delle informazioni sono fondamentali per garantire la sicurezza e l'efficacia delle cure.

Personalmente, ritengo che l'utilizzo di modelli di linguaggio avanzati come quelli descritti sia estremamente promettente per il miglioramento dei servizi sanitari. La capacità di automatizzare compiti come l'analisi e il riassunto di dati può liberare tempo prezioso per il personale medico, consentendo loro di concentrarsi su attività ad alto valore aggiunto come la diagnosi e il trattamento dei pazienti.

Tuttavia, è importante anche considerare i limiti e le sfide associate a questa tecnologia, come la privacy dei dati e la necessità di garantire la sicurezza e l'accuratezza delle informazioni. È fondamentale trovare un equilibrio tra l'adozione di soluzioni innovative e la protezione dei diritti e della sicurezza dei pazienti.

Il codice sorgente di Medical-AI è disponibile su GitHub al seguente link: <https://github.com/grauso-t/medical-ai.git>.

Elenco delle figure

2.1	Model Providers. Fonte: haystack.deepset.ai	10
2.2	Basic RAG. Fonte: llamaindex.ai	13
2.3	Il Modello I/O di LangChain. Fonte: langchain.com	15
2.4	Data Connection. Fonte: langchain.com	17
3.1	LLM-Training	23
3.2	Punteggio MMUL. Fonte: paperswithcode.com [8]	27
4.1	Il Modello MVC.	37
4.2	Architettura	38
4.3	Impostazioni per l’Inferenza Locale.	40
4.4	Prestazioni di Mistral 7B e di diversi modelli di Llama. Fonte: mistral.ai [21]	42
4.5	Prompt per la generazione di query FHIR.	44
4.6	Architettura di Medical-AI	45
4.7	Prompt per l’esecuzione locale di Mistral-7B.	46
4.8	Schermata accesso a Medical-AI	47
4.9	Errore nell’accesso a Medical-AI	48
4.10	Dashboard di Medical-AI	48
4.11	Profilo Personale in Medical-AI	49
4.12	Assistente Virtuale di Medical-AI	50
4.13	Esempio di Risposta.	50
4.14	Esempio paziente.	51
4.15	Esempio di Generazione di un Grafico	52
4.16	Altri Esempi.	53
4.17	Valutazione Percentuale di Somiglianza.	54
4.18	Classe Patient.	56
4.19	Punteggio Traduzione - Patient John Doe	57
4.20	Classe AllergyIntolerance.	58
4.21	Punteggio Traduzione - Allergy Intolerance	59
4.22	Classe Appointment.	60

4.23	Punteggio Traduzione - Appointment	61
4.24	Classe Observation.	62
4.25	Punteggio Traduzione - Observation	63

Elenco delle tabelle

2.1	Quando utilizzare gli agenti. Fonte: langchain.com	18
3.1	Benchmark valutazione prestazioni LLM. Fonte: medium.com	26
4.1	Benchmark valutazione prestazioni LLM locali.	40

Elenco degli acronimi

AI Intelligenza Artificiale. 3

AIML Artificial Intelligence Markup Language. 4

EHR Informazioni Sanitarie Elettroniche. 3

FHIR Framework for the Interoperability of Health Information Resources. 1

GQA Grouped Query attention. 42

JSON JavaScript Object Notation. 1

LLaMa Local Large Memory Access. 27

LLM Large Language Models. 1

NLP Natural Language Processing. 5

RAG Retrieval Augmented Generation. 16

RPA Robotic Process Automation. 5

SWA Sliding Window Attention. 42

Glossario dei termini

Fine-Tuning: Riaddestramento di un modello che è stato già addestrato, avrà quindi tutta una serie di logiche già ben definite e comprovate, per la nuova classificazione..

27

Input Multimodale: Tipo di input che comprende più modalità di input diverse, come testo, immagini, suoni, video, ecc.. 15

Parsing dei Dati: Processo di analisi e interpretazione di un insieme di dati. Consente di estrarre informazioni rilevanti da dati grezzi, rendendoli utilizzabili per il software o l'applicazione in uso.. 49

Prompt: Costituisce un testo in linguaggio naturale che guida l'AI generativa a eseguire una specifica attività.. 43

Python Bindings: Una tecnica che consente al linguaggio di programmazione Python di utilizzare funzionalità scritte in un altro linguaggio. 9

Sentiment: Nel contesto dell'analisi del testo o dell'elaborazione del linguaggio naturale si riferisce alla valutazione del tono emotivo o dell'atteggiamento espresso nel testo..
22

Tokenizzazione: Processo di suddivisione di un testo in unità più piccole, chiamate token. Questi token possono essere parole, numeri, punteggiatura o altri simboli significativi.. 23

Bibliografia

- [1]Prashansa Agrawal. Artificial intelligence in drug discovery and development. Journal of Pharmaceutical and Biomedical Analysis, 6, 2018.
- [2]Amazon Web Services (AWS). What is a large language model? <https://aws.amazon.com/what-is/large-language-model/>, n.d. Accessed: March 26, 2024.
- [3]Yuvanesh Anand, Zach Nussbaum, Brandon Duderstadt, Benjamin Schmidt, and Andriy Mulyar. Gpt4all: Training an assistant-style chatbot with large scale data distillation from gpt-3.5-turbo. <https://github.com/nomic-ai/gpt4all>, 2023.
- [4]Rajdeep Biswas. Valutazione di modelli linguistici di grandi dimensioni (llm): un insieme standard di metriche per una valutazione accurata. Articolo pubblicato su LinkedIn il 13 dicembre 2023.
- [5]Gianluigi Bonanomi. Guida a gpt4all, 2023.
- [6]Samuel R. Bowman. Eight things to know about large language models. arXiv, 2023.
- [7]Harrison Chase. Langchain.
- [8]Papers With Code. Multi-task language understanding on mmlu.
- [9]Emily Rosemary Collins. GPT4All vs. Llama: Unveiling the AI Showdown, June 27 2023.
- [10]Gerganova. llama-cpp-python.
- [11]Gerganova. Llama.cpp.
- [12]Red Hat. Cos'è l'orchestrazione?, 2019.
- [13]Red Hat. What are large language models, n.d.
- [14]Health Level Seven International. Fast healthcare interoperability resources (fhir), 2019.

-
- [15]Jane Huang. Evaluating llm systems: Metrics, challenges, and best practices, March 2024. Medium.
- [16]Zoumana Keita. Llama.cpp tutorial: A complete guide to efficient llm inference and implementation, 2023.
- [17]Yikuan Li, Hanyin Wang, Halid Z. Yerebakan, Yoshihisa Shinagawa, and Yuan Luo. Fhir-gpt enhances health interoperability with large language models, 2023.
- [18]Yikuan Li, Hanyin Wang, Halid Z. Yerebakan, Yoshihisa Shinagawa, and Yuan Luo. Fhir-gpt enhances health interoperability with large language models, 2023.
- [19]Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. <https://huggingface.co/Open-Orca/OpenOrca>, 2023.
- [20]Jerry Liu. LlamaIndex, 11 2022.
- [21]Mistral-7B. Announcing mistral 7b. <https://mistral.ai/news/announcing-mistral-7b/>, September 2023. Accessed on April 14, 2024.
- [22]OpenAI. Better Language Models and Their Implications. <https://openai.com/research/better-language-models/>, February 2019. Archived from the original on 2020-12-19. Retrieved 2019-08-25.
- [23]OpenAI. Gpt-3.5: A state-of-the-art language model. Journal of Artificial Intelligence Research, 12(3):45–63, 2022.
- [24]Michael Pietsch, Tushar Soni, Boris Chan, Timo Möller, and Bojan Kostić. Haystack. GitHub, 2020. <https://github.com/deepset-ai/haystack/>.
- [25]Surya Roca, Jorge Sancho, José García, and Álvaro Alesanco. Microservice chatbot architecture for chronic patient support. Journal of Biomedical Informatics, 102:103305, 2020.
- [26]The AI Team @ Replit. Replit: Collaborative Programming. <https://replit.com/>, April 2023. Accessed on April 18, 2023.
- [27]Wikipedia contributors. Electronic health record — Wikipedia, the free encyclopedia, 2024. [Online; accessed 21-April-2024].
- [28]Wikipedia contributors. Fast healthcare interoperability resources — Wikipedia, the free encyclopedia, 2024. [Online; accessed 21-April-2024].

-
- [29]Wikipedia contributors. Health information exchange — Wikipedia, the free encyclopedia, 2024. [Online; accessed 21-April-2024].
- [30]Wikipedia contributors. Llama — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=LLaMA&oldid=1212315599>, 2024. [Online; accessed 6-April-2024].
- [31]Wikipedia contributors. Picture archiving and communication system — Wikipedia, the free encyclopedia, 2024. [Online; accessed 21-April-2024].

Ringraziamenti

Desidero esprimere la mia gratitudine al Professore Gennaro Costagliola, mio relatore di tesi. Senza il suo sostegno, la sua guida e la sua pazienza, questo lavoro non sarebbe stato possibile.

Un sincero ringraziamento va ai miei genitori, la cui costante presenza, il loro incoraggiamento e l'amore incondizionato sono stati fondamentali per il raggiungimento di questo traguardo. Il vostro sostegno, la vostra pazienza e i vostri sacrifici sono stati essenziali per arrivare fin qui.

Ringrazio i miei coinvilini e amici, Cristian, Emilio e Francesco, per aver condiviso con me non solo gli spazi della nostra casa, ma anche i momenti di gioia, sconforto e crescita personale. Un ringraziamento speciale va a Giovanni, un amico affidabile e generoso. Grazie per esserci sempre, pronto per una chiacchierata o un consiglio.

Ringrazio gli amici della Biblioteca Scientifica per il loro continuo sostegno. In particolare, desidero ringraziare Andrea e Gennaro, il cui supporto è stato cruciale durante il mio percorso di studi. Ringrazio Mariantonietta, che con la sua presenza e le sue innumerevoli chiacchiere è stata un faro nei momenti difficili. La vostra amicizia ha reso più piacevoli le lunghe sessioni di studio.

Ringrazio i compagni Domenico, Errico, Francesco, Felice, Giuseppe e a tutti gli amici di "Classe 3 Resto 2". La vostra preziosa amicizia ha reso questo percorso più sereno e piacevole. Grazie a tutti voi per aver condiviso con me questi momenti indimenticabili.

Un ringraziamento speciale va a Veronica per la sua presenza nella mia vita, per il suo costante sostegno e per tutti i momenti condivisi che hanno reso ogni istante unico. Grazie per essere sempre al mio fianco, anche da lontano. La tua amicizia è un tesoro prezioso che porterò sempre con me.



Adesso si ricomincia...