UNIVERSITÀ
DI PARMA

Dipartimento di Scienze Matematiche, Fisiche e Informatiche

Corso di Laurea Triennale in Informatica

IA Generativa: Applicazioni e Impatti dei Large Language Models

Generative AI: Impacts and Applications of Large Language Models

Candidato:
Alessandro Domenico Salvatore

Relatore:
Prof. Nome Cognome

Anno Accademico 2023–2024

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Negli ultimi anni, l'esplosione dei Large Language Models (LLMs) ha comportato significativi cambiamenti, sia positivi che negativi. L'obiettivo principale di questa ricerca è valutare l'impatto che i modelli di linguaggio hanno avuto sulla società e fornire una panoramica delle loro diverse applicazioni. Questa analisi permette di comprendere come i modelli di linguaggio possano rappresentare uno strumento utile in ambiti specifici, come ad esempio nella medicina, dove possono migliorare l'efficienza e l'accesso alle informazioni. Tuttavia, è altrettanto importante riconoscere le problematiche associate al loro utilizzo, comprese le potenziali distorsioni, malintesi e abusi che possono derivare da un loro impiego improprio. In sintesi, la ricerca mira a evidenziare sia i benefici che i rischi legati ai Large Language Models, promuovendo una riflessione critica su come questi strumenti possano essere utilizzati in modo responsabile ed efficace.
Nel primo capitolo viene fornita una panoramica sull'Intelligenza Artificiale Generativa, tracciando la sua evoluzione storica fino ai modelli generativi più recenti, sottolineando in modo particolare i modelli di diffusione e i transformers.
Il secondo capitolo esplora esempi concreti di Large Language Models, esaminando le loro applicazioni nel mondo del lavoro e le problematiche associate al loro utilizzo. Viene inoltre discussa una serie di tecniche progettate per affrontare e risolvere le problematiche legate alla generazione.
Nel terzo capitolo viene presentato un progetto di chatbot sviluppato durante l'esperienza di tirocinio curriculare. Questo progetto illustra l'implementazione di un'architettura Retrieval-Augmented Generation (RAG), dimostrando le potenzialità e le applicazioni pratiche della tecnologia.

Capitolo 1

Intelligenza Artificiale Generativa

1.1 Definizione di Generative AI

La definizione di Intelligenza Artificiale Generativa si riferisce a tutte le tecniche computazionali per generare contenuto come immagini, testo, audio ed altre tipologie di dati partendo da uno specifico training data[Feuerriegel et al., 2024]. Il training data è un'insieme di dati che viene utilizzato per eseguire un addestramento sul modello di rete neurale.
Un modello di IA Generativa è un tipo di architettura di Machine Learning che utilizza algoritmi di IA, attingendo dai pattern e dalle relazioni osservate nel training data[Feuerriegel et al., 2024].

1.2 Breve storia ed evoluzione

L'inizio dell'ascesa dei modelli generativi può essere individuata già nei lavori di Alan Turing. Turing ha contribuito alla nascita dell'Intelligenza Artificiale attraverso il suo articolo del 1950 "**Computing Machinery and Intelligence**" [Turing, 2009], introducendo il Turing Test come uno strumento che misura l'intelligenza di una macchina. Turing ha scritto inoltre un articolo nel 1948, intitolato "**Intelligent Machinery**" [Turing, 2004] , in cui espimeva il concetto di una macchina capace di imparare dall'esperienza. Il concetto risulta interessante perché ci permette di immaginare una macchina capace di apprendere e di essere addestrata, ponendo così le basi del Machine Learning.
Prima del 2014, tutti i modelli di deep learning esistenti, come ad esempio le Convolutional Neural Networks (CNNs)[LeCun et al., 1998], le Recurrent Neural Networks (RNNs)[Hopfield and Tank, 1985] e le Long Short-Term

Memory (LSTM)[Hochreiter and Schmidhuber, 1997], erano principalmente descrittivi. Questi modelli miravano a spiegare i pattern nei dati e a fare previsioni basate sulle informazioni disponibili.[Bengesi et al., 2024]
I GAN (Generative Adversial Network)[Goodfellow et al., 2020] hanno aperto la strada alle Generative AI. Il loro obiettivo principale è di generare nuovi campioni di dati che assomiglino strettamente ai modelli osservati nei dati di addestramento [Brophy et al., 2023][Zhou et al., 2023a].

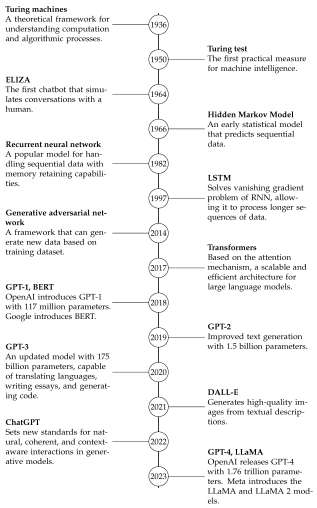


Figura 1.1: Illustrazione della linea del tempo dello sviluppo della Generative AI, dalla Macchina di Turing fino a GPT-4 [Zhang et al., 2023].

1.3 Generative AI sul lavoro

Sin dalla loro introduzione intorno al 1970, i computer si sono imposti prepotentemente nel mondo del lavoro grazie alla loro capacità di eseguire in modo efficiente istruzioni pre-programmate.
La computerizzazione ha notevolmente ridotto i tempi di lavoro specialmente in operazioni di routine, come l'inserimento dei dati, la contabilità e le mansioni nella catena di montaggio, determinando così una diminuzione dei posti di lavoro e dei salari in questi settori [Acemoglu and Autor, 2011]. Allo stesso tempo, questo fenomeno ha incrementato la domanda di lavoratori con abilità di programmazione, analisi dei dati e di ricerca, portando ad un aumento della disuguaglianza salariale soprattutto negli Stati Uniti [Katz and Murphy, 1992].
In contrasto, le generative AI non richiedono specifiche istruzioni per eseguire

le operazioni, ma possiedono la capacità di imparare dalle esperienze ed applicare le regole incosciamente durante il processo. Questa capacità è definita conoscenza tacita e permette di svolgere operazioni che prevedono di fare affidamento sul giudizio e sull'esperienza del modello [Fischer and Mandell, 2009] [Nguyen and Nadi, 2022]. I modelli di diffusione ed i transformers hanno avuto un grande impatto nel mondo del lavoro, trovando diverse applicazioni in vari settori grazie alla loro versatilità.

1.4 Modelli di diffusione

I modelli di diffusione sono una classe di modelli generativi che hanno lo scopo di interpretare distribuzioni di dati molto complesse. Per ottenere uno di questi modelli si devono affrontare due fasi: la diffusione in avanti e la diffusione all'indietro. Il primo processo serve per alterare l'architettura di distribuzione dei dati, aumentando gradualmente il livello del rumore all'interno dei dati di input fino ad ottenere un rumore gaussiano puro. Il processo di diffusione all'indietro effettua l'operazione inversa, fornendo così un modello generativo molto flessibile e capace di generare complesse distribuzioni di dati a partire da un rumore casuale [Shokrollahi et al., 2023].
Per esempio, i Denoinsing, Diffusion Probabilistic Models (DDPMs) appartengono a questa classe di modelli e approssimano i parametri di un processo di diffusione utilizzando l'inferenza variazionale per produrre un'immagine di alta qualità.[Ho et al., 2020]

1.4.1 Applicazioni nell'assistenza sanitaria

L'assistenza sanitaria è il settore in cui si sono verificati i maggiori progressi grazie all'uso dei modelli di diffusione. Le applicazioni possono includere: la ricostruzione di immagini, le traduzione di immagini e la generazione di un'immagine, tra le principali.

Ricostruzione di immagini

La **ricostruzione di immagini** è un processo di conversione di un'insieme di dati di difficile interpretazione in un'immagine target di facile comprensione. Infatti, ottenere immagini di alta qualità è fondamentale nella immagini mediche, non solo per la riduzione dei rischi per i pazienti ma anche per l'abbassamento dei costi [Levac et al., 2023].
Le normali tecniche per ricavare immagini dettagliate ed alta risoluzione prevedono alte dosi e prolungato tempo di esposizione alle radiazioni, come

per esempio nel caso della Tomografia Computerizzata (CT) o la Risonanza Magnetica (MRI). Di conseguenza, sono state sviluppate strategie per ridurre l'esposizione alle radiazioni come l'uso di dosi ridotte o i processi di imaging con campionamento ridotto. Tuttavia, queste tecniche possono ridurre il rapporto segnale-rumore (SNR) e il rapporto contrasto-rumore (CNR) influenzando la qualità finale dell'immagine. Questo problema può essere risolto applicando la ricostruzione dell'immagine [Shokrollahi et al., 2023].
Nel 2022 è stato introdotto un approccio chiamato MC-DDPM utilizzando il framework DDPM per raffinare la ricostruzione di immagini mediche sottocampionate. MC-DDPM è un approccio che potrebbe essere utilizzato anche per la ricostruzione CT e PET [Xie and Li, 2022].
Nel 2023 per la ricostruzione delle immagini MRI è stata introdotta una tecnica chiamata **Adaptive Diffusion Priors** (AdaDiff). Il metodo regola in modo dinamico i suoi priors durante la fase di inferenza per allinearsi meglio con la distribuzione dei dati di test migliorando la qualità dell'immagine [Güngör et al., 2023].

Traduzione di immagini

La **traduzione da immagine ad immagine** è invece una tecnica che permette di trasformare un'immagine di un tipo in un'altra immagine di tipo diverso. Infatti, spesso, per ottenere una corretta diagnosi, dopo un CT è necessario effettuare anche un MRI, rendendo il processo più lungo e costoso. Con l'utilizzo dei modelli di diffusione si può ridurre il tempo di esposizione grazie al miglioramento della flessibilità e della completezza dell'imaging medico. Le due architetture fondamentali per la traduzione di immagini sono: pix2pix[Isola et al., 2017] e CycleGAN[Chu et al., 2017].
Nel 2023 è stato impiegato un modello di diffusione chiamato SynDiff, che ha dimostrato un'elevata efficacia nella traduzione non supervisionata di immagini mediche. Dopo numerosi test, è stato dimostrato che questo metodo ottiene buoni risultati in termini di precisione, riduzione degli artefatti ed efficienza computazionale rispetto ad altri modelli. Le immagini generate risultano indistinguibili rispetto a quelle originali [Özbey et al., 2023].

Generazione di immagini

La generazione di immagini è stata una delle prime applicazioni all'interno della sanità, soprattutto per la fabbricazione di immagini mediche artificiali 2D/3D. Medfusion è un DDPM condizionale latente, utilizzato per la sintesi di immagini mediche. Nel lavoro di [Müller-Franzes et al., 2023] è stata condotta un'analisi approfondita di dataset comprendenti immagini RGB del

fondo oculare, scansioni CT del torace e scansioni MRI 3D del cervello di diversi soggetti. Gli autori hanno impiegato diverse metriche di valutazione della qualità delle immagini, tra cui il Peak Signal-to-Noise Ratio (PSNR), il Structural Similarity Index (SSIM) e la Fréchet Inception Distance (FID). I risultati della loro ricerca hanno dimostrato che l'approccio adottato offre una maggiore stabilità e interpretabilità, nonché una qualità superiore delle immagini rispetto ai modelli Generative Adversarial Networks.

1.5 Transformers

1.5.1 Definizione di Transformers

I Generative Transformers (GT) sono un'estensione delle reti neurali e sono designati a pesare il significato delle diverse parti di una sequenza di input, catturando le relazioni intricate tra i dati. Con un corretto addestramento ed il giusto contesto i GT possono generare ogni tipo di dato [Zhang et al., 2023].

1.5.2 Vantaggi dei Transformers rispetto alle CNNs e RNNs nell'Intelligenza Artificiale

Questi modelli sono stati originariamente progettati per lo sviluppo del Natural Language Processing (NLP), ma successivamente sono stati adattati e applicati anche ad altri ambiti, come ad esempio la Computer Vision [Han et al., 2022]. I transformers, grazie alla loro architettura avanzata e alla capacità di catturare relazioni a lungo raggio nei dati, presentano diversi vantaggi rispetto alle architetture CNNs ed RNNs. In primis, consentono un'elaborazione parallela dei dati e migliorano significativamente l'efficienza computazionale traducendosi in un risparmio di costi e risorse. Inoltre i GT sono in grado di raggiungere minimi locali o globali con un numero inferiore di passi di addestramento. Questo li rende particolarmente efficaci per compiti complessi come per l'elaborazione dei dati del linguaggio naturale. A differenza delle CNNs e delle RNNs, i transformers non soffrono di problemi legati alle dipendenze a lungo raggio, un limite significativo delle RNNs, che spesso faticano a mantenere informazioni rilevanti su sequenze di dati estese. Inoltre, i transformers evitano il problema della scomparsa o dell'esplosione del gradiente, che può ostacolare il processo di apprendimento nelle reti neurali tradizionali. Questi vantaggi rendono i transformers una scelta preferibile per molte applicazioni di intelligenza artificiale, dalla traduzione automatica al riconoscimento del linguaggio [Vaswani et al., 2017].

1.5.3 Architettura del modello

Encoder-Decoder Structure

I transformers possiedono una struttura encoder-decoder come rappresentato nella figura 1.2. L'encoder mappa una rappresentazione simbolica di sequenza $(x_1, \ldots, x_n)$ in una rappresentazione continua di sequenza z=$(z_1, \ldots, z_n)$. È composto da una pila di 6 livelli identici, ognuno dei quali possiede due sottolivelli seguiti da un livello di normalizzazione. Il primo sottolivello è il meccanismo di multi-head self-attention, mentre il secondo è una normale rete feed-forward completamente connessa. Il decoder, dato z, genera una sequenza di simboli $(y_1, \ldots, y_n)$, un elemento per volta, che poi saranno consumati come input addizionale per la prossima generazione. La sua struttura è pressoché simile a quella dell'encoder, con ogni livello che possiede un sottolivello aggiuntivo che esegue il multi-head attention sull'output della pila dell'encoder. Il meccanismo di self-attention viene modificato con una maschera speciale per impedire che le posizioni future possano influenzare le posizioni attuali, quindi la previsione per y_i non dipende dalle posizioni future $y_{i+1}, y_{i+2}, \ldots$, e il contesto prevede solo i simboli già generati fino a quel punto [Vaswani et al., 2017].

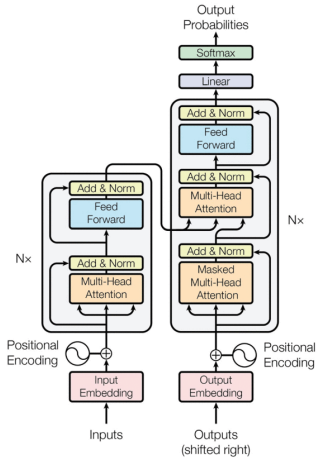


Figura 1.2: Transformer - architettura del modello [Vaswani et al., 2017].

Il Meccanismo Attention

É un meccanismo che consente di capire il contesto della parola estrapolandolo dalla parte più significativa dell'input. La funzione attention può essere rappresentata come una mappatura tra una query ed un'insieme di coppie chiave valore e l'output. L'output è la somma pesata di tutti i valori, dove

il peso di ogni valore è calcolato dalla funzione di compatibilità della query con la chiave corrispondente. La funzione di attention permette di calcolare simultaneamente un'insieme di query, nella formula 6 viene determinato l'output della funzione.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

Dove Q rappresenta l'insieme di query, K l'insieme delle chiavi nella matrice e V l'insieme dei valori nella matrice. Per evitare che la funzione Softmax sia indirizzata verso zone di gradiente estremamente piccole a causa di un fattore di scala molto grande, nella formula 6 Il fattore di scala d_k viene sostituito con $\frac{1}{d_k}$ [Vaswani et al., 2017].

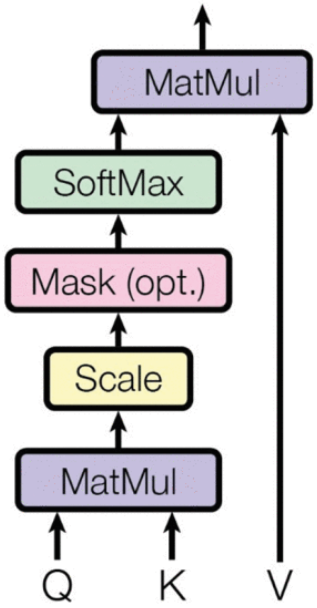


Figura 1.3: Scaled Dot-Product Attention [Vaswani et al., 2017]

Multi-Head Attention permette al modello di imparare diversi significati della sequenza di input, in questo modo il meccanismo di self-attention può essere eseguito molteplice volte in parallelo. Ogni esecuzione parallela è chiamata testa. Ogni testa esplora l'input attraverso una diversa prospettiva o "sotto-spazio di rappresentazione", permettendo al modello di apprendere diverse relazioni tra gli elementi. La formula 7 mostra come gli attention output indipendenti sono concatenati e linearmente trasformati nella dimensione aspettata [Vaswani et al., 2017].

$$MultiHead\,(Q, K, V) = concat\,(head_1, \ldots head_h)\,W^o \tag{7}$$

dove $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

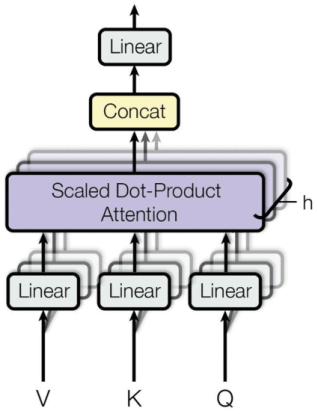


Figura 1.4: Scaled Dot-Product Multi-Head Attention [Vaswani et al., 2017].

Capitolo 2

Large Language Models (LLMs)

2.1 Introduzione e caratteristiche degli LLMs

Un modello di linguaggio di grandi dimensioni (Large Language Model, LLM) è un tipo di modello di intelligenza artificiale caratterizzato da un numero estremamente elevato di parametri e addestrato su un vasto corpus di dati testuali. Questi modelli sono progettati per eseguire operazioni linguistiche con un elevato grado di accuratezza.
L'architettura Transformer ha giocato un ruolo cruciale nel successo di questi modelli. Il meccanismo di self-attention, che costituisce la base di questa architettura, consente di valutare l'importanza relativa delle diverse parti dell'input e di comprendere le relazioni tra le parole in modo più efficace rispetto alle architetture precedenti.
Un ulteriore elemento chiave del successo degli LLM è rappresentato dal processo di pre-addestramento. Questo processo utilizza un vasto insieme di dati, che può includere articoli, libri e altre forme di testo, per formare il modello a eseguire operazioni linguistiche con maggiore competenza. Il pre-addestramento su dati di larga scala permette al modello di acquisire una comprensione approfondita delle strutture linguistiche e delle conoscenze generali, che vengono poi affinati per compiti specifici attraverso un processo di addestramento mirato [Kasneci et al., 2023].

2.2 Esempi di LLMs attuali

2.2.1 Panoramica dei modelli LLMs

La tabella 2.1 presenta diversi modelli di linguaggio e le loro rispettive caratteristiche. La colonna "tunability" indica i modelli che possono essere sottoposti a fine-tuning per compiti specifici, al fine di migliorarne l'efficacia per tali compiti [Yao et al., 2024]. Tra i modelli, si possono osservare le diverse versioni di GPT, che nel tempo hanno ottenuto un successo particolare.

Modello	Data	Azienda	Open-Source	Parametri	Tunability
GPT-4 [Achiam et al., 2023]	2023.03	OpenAI	NO	1.7 T	NO
GPT-3.5-turbo	2021.09	OpenAI	NO	175 B	NO
Cohere-medium [Liang et al., 2022]	2022.07	Cohere	NO	6 B	YES
Cohere-large [Liang et al., 2022]	2022.07	Cohere	NO	13 B	YES
Cohere-xlarge [Liang et al., 2022]	2022.07	Cohere	NO	52 B	YES
BERT [Devlin et al., 2018]	2018.08	Google	YES	340 M	YES
T5 [Raffel et al., 2020]	2019	Google	YES	11 B	YES
PaLM [Narang and Chowdhery, 2022]	2022.04	Google	YES	540 B	YES
LLaMA [Meta, 2023]	2023.02	Meta AI	YES	65 B	YES
CTRL [Keskar et al., 2019]	2019	Salesforce	YES	1.6 B	YES
Dolly 2.0	2023.04	Databricks	YES	12 B	YES

Tabella 2.1: Confronto tra i diversi modelli di linguaggio [Yao et al., 2024]

2.2.2 Generative Pre-trained Transformer (GPT)

GPT è un modello di linguaggio autoregressivo che genera sequenze di parole, codice ed altre forme di dato, partendo da una sorgente di input, chiamata prompt. Il modello è stato addestrato utilizzando il supercomputer Microsoft Azure AI, con costi stimati intorno ai 12 milioni di dollari. La prima iterazione risale al 2018 e ha impiegato 110 milioni di parametri. L'anno successivo, GPT-2 ha utilizzato 1,5 miliardi di parametri, mentre GPT-3 ne utilizza infine 175 miliardi [Floridi and Chiriatti, 2020]. Il modello più recente, GPT-4, è stato addestrato con 1 trilione di parametri, e OpenAI sta già lavorando allo sviluppo di GPT-5.

GPT-4 Performance

La **perdita finale** di un modello può essere approssimata utilizzando leggi di scala, in questo modo riusciamo a prevedere la relazione tra le risorse computazionali impiegate e la performance del modello [Achiam et al., 2023]. Nella figura 2.1, è stata analizzata la perdita finale di GPT-4 utilizzando un

dataset molto ampio di codici non incluso nell'insieme di addestramento, impiegando la legge di scala 2.1 [Henighan et al., 2020].

$$L(C) = aC^b + c \tag{2.1}$$

Dove **a** è una costante che determina l'ampiezza dell'effetto della variabile **C**, che rappresenta una misura della risorsa computazionale o della dimensione del modello, come il numero di parametri o la dimensione del dataset. **b** è l'esponente che indica come la perdita dipende dalla variabile C, ed infine **c** è una costante aggiuntiva che rappresenta un termine di perdita asintotica o residua. La previsione è basata utilizzando modelli addestrati nello stesso modo, ma utilizzando 10.000 volte meno calcolo di GPT-4 come mostrato nella figura 2.1 [Achiam et al., 2023].

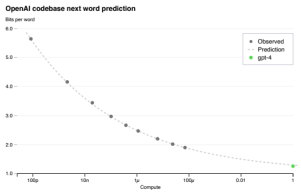


Figura 2.1: Performance di GPT-4 e modelli più piccoli. La metrica è la perdita finale su un dataset interno, la linea tratteggiata corrisponde alle legge adattata ai modelli più piccoli, predicendo con precisione la perdita finale di GPT-4. L'asse x rappresenta il calcolo di addestramento normalizzato in modo che GPT-4 sia uguale a 1 [Achiam et al., 2023].

È stato utilizzato il dataset HumanEval [Chen et al., 2021] per valutare metriche di capacità, come ad esempio il tasso di successo. Il test prevedeva di misurare la sintesi di funzioni Python di varia complessità. La previsione del tasso di successo su un sottoinsieme del dataset HumanEval è stata effettuata estraendo dati da modelli addestrati con al massimo 1.000 volte meno calcolo. Poiché la performance potrebbe deteriorarsi con la scala per un singolo problema nel dataset, la relazione approssimativa di legge di potenza 2.2 cerca di affrontare questo problema [Achiam et al., 2023].

$$-\mathbb{E}_P\left[\log(\text{pass\_rate}(C))\right] = \alpha C^{-k} \tag{2.2}$$

Dove k e α sono costanti positive e P rappresenta il sottoinsieme dei problemi presi da HumanEval dataset. L'obiettivo era prevedere la performance di GPT-4 sul dataset prima che l'addestramento fosse completato, basandosi sui risultati dei modelli più piccoli. I problemi sono stati suddivisi dai modelli più piccoli in sei categorie di difficoltà. La figura 2.2 mostra i risultati della

categoria numero 3, dove era possibile stimare con precisione il logaritmo del tasso di successo utilizzando i modelli più piccoli; anche le altre categorie hanno comunque ottenuto prestazioni molto buone. GPT-4 ha sottoperformato nelle categorie più facili, mentre ha ottenuto risultati migliori nelle categorie che contenevano classi di problemi più difficili [Achiam et al., 2023].

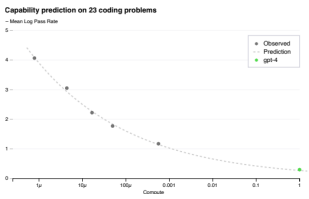


Figura 2.2: Performance di GPT-4 e modelli più piccoli. La metrica è la media del logaritmo del tasso di successo su un sottoinsieme di HumanEval. L'asse x rappresenta il calcolo di addestramento normalizzato in modo che GPT-4 sia pari a 1 [Achiam et al., 2023].

GPT-4 è stato testato su esami umani senza addestramento specifico per questi. Le domande includevano vari formati (scelta multipla e risposta libera) e sono stati utilizzati prompt specifici per ciascun tipo. I punteggi sono stati calcolati combinando i risultati delle diverse tipologie di domande e i percentili sono stati stimati e riportati nella figura 2.3 [Achiam et al., 2023].

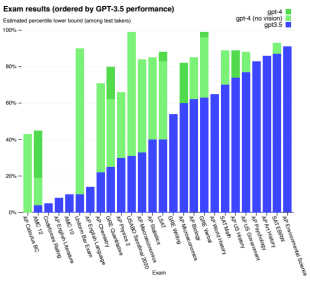
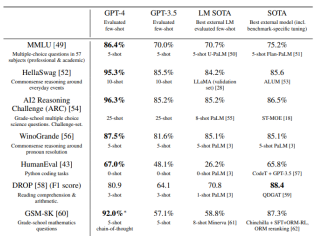


Figura 2.3: performance di GPT-4 su esami accademici e professionali, simulando le condizioni e il sistema di valutazione degli esami reali [Achiam et al., 2023].

Nella figura 2.4 si possono osservare le prestazioni di GPT-4 sui benchmark tradizionalmente utilizzati per valutare i modelli linguistici di grandi dimensioni, ottenendo infine ottimi risultati in confronto ad essi [Achiam et al., 2023].



	GPT-4 Evaluated few-shot	GPT-3.5 Evaluated few-shot	LM SOTA Best external LM evaluated few-shot	SOTA Best external model (incl. benchmark-specific tuning)
MMLU [49] Multiple-choice questions in 57 subjects (professional & academic)	**86.4%** 5-shot	70.0% 5-shot	70.7% 5-shot U-PaLM [50]	75.2% 5-shot Flan-PaLM [51]
HellaSwag [52] Commonsense reasoning around everyday events	**95.3%** 10-shot	85.5% 10-shot	84.2% LLaMA (validation set) [28]	85.6 ALUM [53]
AI2 Reasoning Challenge (ARC) [54] Grade-school multiple choice science questions. Challenge-set.	**96.3%** 25-shot	85.2% 25-shot	85.2% 8-shot PaLM [55]	86.5% ST-MOE [18]
WinoGrande [56] Commonsense reasoning around pronoun resolution	**87.5%** 5-shot	81.6% 5-shot	85.1% 5-shot PaLM [3]	85.1% 5-shot PaLM [3]
HumanEval [43] Python coding tasks	**67.0%** 0-shot	48.1% 0-shot	26.2% 0-shot PaLM [3]	65.8% CodeT + GPT-3.5 [57]
DROP [58] (F1 score) Reading comprehension & arithmetic.	80.9 3-shot	64.1 3-shot	70.8 1-shot PaLM [3]	**88.4** QDGAT [59]
GSM-8K [60] Grade-school mathematics questions	**92.0%*** 5-shot chain-of-thought	57.1% 5-shot	58.8% 8-shot Minerva [61]	87.3% Chinchilla + SFT+ORM-RL, ORM reranking [62]

Figura 2.4: prestazioni di GPT-4 su benchmarks accademici [Achiam et al., 2023].

La maggior parte dei ML benchmarks sono scritti in inglese, quindi gli MMLU benchmarks[1] sono stati tradotti in altre lingue e testati. Nella figura 2.5 possiamo osservare i risultati dei test tra GPT-4 e gli altri modelli di linguaggio. GPT-4 ha superato di gran lunga le prestazioni del modello precedente e di tutti gli altri modelli, sia in inglese che in tutte le altre lingue [Achiam et al., 2023].

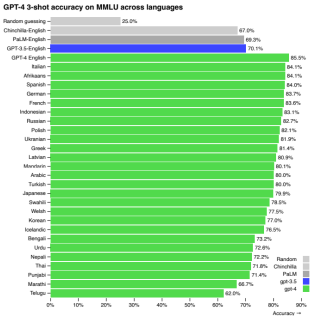


Figura 2.5: prestazioni di GPT-4 sui MMLU benchmarks tradotti in altre lingue in confronto ad altri modelli [Achiam et al., 2023].

In conclusione, nonostante i risultati eccezionali ottenuti da GPT-4, il modello non è ancora completamente affidabile, poiché può generare fatti inesatti ("allucinazioni") e commettere errori di ragionamento. È quindi necessario prestare sempre attenzione e fare un uso attento e consapevole dello strumento [Achiam et al., 2023].

[1] un'insieme di problemi a scelta multipla di 57 discipline diverse

2.2.3 Google Gemini

Gemini è il modello multimodale nativo[2] sviluppato da Google. La capacità di processare insiemi di dati complessi, come grafici e immagini, consente a questo modello di trovare applicazione in ambiti come la medicina e l'oftalmologia. In medicina, oltre alla sua abilità nell'analisi delle immagini, Gemini è in grado di comprendere e interpretare la letteratura medica, lo storico dei pazienti e i dati di ricerca. In oftalmologia, Gemini può diagnosticare condizioni oculari, analizzare i sintomi riportati dai pazienti e suggerire piani di trattamento basati su ricerche recenti e linee guida cliniche [Masalkhi et al., 2024]. Al contrario, modelli di linguaggio come ChatGPT possono avere difficoltà nella comprensione del contesto delle informazioni o nel fornire dati aggiornati, rendendo complesso l'utilizzo di queste tecnologie in ambiti specialistici [Waisberg et al., 2023a] [Kocoń et al., 2023] [Jeyaraman et al., 2023] [Waisberg et al., 2023b].

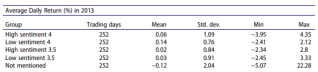
2.3 Impatti e Applicazioni degli LLMs

2.3.1 Applicazioni nel business e nella finanza

Nei recenti studi di [Chen et al., 2023a] è stato condotto un esperimento per dimostrare l'abilità di ChatGPT nel condurre un'analisi del sentiment nel mercato finanziario. Nel 2013, in California, è entrato in vigore il programma cap-and-trade per ridurre le emissioni di gas serra. Sono state esaminate 321 aziende con sede in California, Nevada, Oregon e Arizona, utilizzando i rapporti annuali delle aziende successivi all'approvazione della politica nel 2011. È stato fornito a ChatGPT il seguente prompt template per attribuire un punteggio al sentiment negativo: *'On a probability scale of 0 to 1, what is the negative sentiment score of the discussion for capand-trade policy in the provided text:' + 'searched text' + 'Please only output the sentiment score without any narrative nor analysis. So the sentiment score of this text is: '* La figura 2.6 mostra come le aziende con punteggi di sentiment negativo più alti abbiano registrato ritorni giornalieri più bassi (0.06) rispetto a quelle con punteggi di sentiment negativo più bassi (0.14). Inoltre, le aziende che non

[2]Modello di intelligenza artificiale progettato per comprendere e generare dati attraverso diversi tipi di input e output, come testo, immagini, audio e video, in modo nativo. Questo significa che il modello è stato concepito e addestrato fin dall'inizio per lavorare con più modalità di dati, piuttosto che essere un modello originariamente unimodale (ad esempio solo testuale) che è stato successivamente adattato per gestire altre modalità.

hanno menzionato le parole chiave hanno avuto i valori di ritorno più bassi e fluttuazioni maggiori nei ritorni (-0.12 e deviazione standard di 2.04).



Average Daily Return (%) in 2013

Group	Trading days	Mean	Std. dev.	Min	Max
High sentiment 4	252	0.06	1.09	−3.95	4.35
Low sentiment 4	252	0.14	0.76	−2.41	2.12
High sentiment 3.5	252	0.02	0.84	−2.34	2.8
Low sentiment 3.5	252	0.03	0.91	−2.45	3.33
Not mentioned	252	−0.12	2.04	−5.07	22.28

Figura 2.6: Valori di ritorno per ogni categoria del sentiment negativo generato da GPT-3.5 E GPT-4 [Chen et al., 2023a].

Il case study ha dimostrato che dopo aver introdotto una politica aziendale, il sentiment negativo delle aziende, rilevato da ChatGPT, ha predetto una migliore capacità di gestione del rischio e una performance azionaria meno volatile. Il sentiment negativo generato da ChatGPT può indicare efficacemente i fattori di rischio. Nel settore finanziario è fondamentale considerare le limitazioni dei modelli utilizzati. I dati generati da un modello potrebbero condurre ad addestramenti errati dei modelli stessi, decisioni fuorvianti e significative perdite economiche. Inoltre, è importante tenere presente che le predizioni si basano su dati storici e non sono in grado di prevedere eventi inaspettati. Un ulteriore limite risiede nella variabilità delle risposte: domande con lo stesso significato, ma formulate in modo diverso, possono ottenere risposte differenti.

2.3.2 Applicazioni e benefici nella medicina

Gli studi condotti da [Zheng et al., 2024] hanno messo in luce i molteplici vantaggi derivanti dall'impiego dei modelli linguistici di grandi dimensioni nel campo medico. Tra questi, spiccano la capacità di migliorare la diagnosi e la previsione delle patologie, l'integrazione della conoscenza e l'accesso in tempo reale alle informazioni, lo sviluppo di trattamenti personalizzati e di farmaci, la gestione ottimizzata dei pazienti e dei processi sanitari, nonché il potenziamento dell'educazione clinica e la diffusione della conoscenza medica.

Capacità di diagnosi e previsione migliorate

Gli LLMs facilitano la diagnosi precoce e la previsione delle malattie, consentendo interventi tempestivi e personalizzati. Essi sono anche in grado di prevedere l'efficacia dei trattamenti basati su dati clinici e genomici.

Integrazione della conoscenza e accesso in tempo reale

Gli LLMs aggiornano costantemente il corpus delle conoscenze mediche globali, supportando decisioni cliniche basate sulle più recenti evidenze e migliorando la precisione sia diagnostica che terapeutica.

Trattamenti personalizzati e sviluppo di farmaci

Gli LLMs sono in grado di elaborare piani terapeutici su misura per i singoli pazienti e accelerano il processo di sviluppo di nuovi farmaci, prevedendone l'efficacia e i possibili effetti collaterali.

Gestione dei pazienti e ottimizzazione dei processi sanitari

Gli LLMs contribuiscono alla personalizzazione della gestione della salute dei pazienti e al miglioramento dell'efficienza dei processi sanitari, attraverso l'identificazione e l'ottimizzazione dei punti critici nei flussi di lavoro.

Educazione clinica e diffusione della conoscenza medica

Gli LLMs rivestono un ruolo fondamentale nell'ambito dell'educazione medica, simulando scenari clinici realistici e fornendo applicazioni sanitarie che offrono consigli personalizzati per la prevenzione e la gestione delle malattie, aiutando così i pazienti a prendere decisioni informate riguardo alla propria salute.

Applicazioni e Supporto degli LLM in Diverse Specialità Mediche

Gli LLM medici hanno ottenuto notevole successo nei seguenti settori: odontoiatria [Huang et al., 2023], radiologia [D'Antonoli et al., 2024], medicina nucleare [Wang et al., 2023b], clinica [Singhal et al., 2023] e progettazione dei farmaci [Chakraborty et al., 2023]. Inoltre, hanno fornito supporto in ambiti quali la medicina interna [Omiye et al., 2024], la chirurgia [Puladi et al., 2023], l'ostetricia e ginecologia [Grünebaum et al., 2023], le malattie infettive [Schwartz et al., 2024], la medicina genetica [Feldman et al., 2019] e le malattie croniche [Biswas, 2023].
Gli LLM medici sono addestrati su ampi corpus di dati biomedici, BioGPT [Luo et al., 2022] e BioBERT [Lee et al., 2020] sono esempi rappresentativi di tali modelli. BioBERT, rispetto ad altri modelli, ha dimostrato di migliorare del 0,62% il punteggio F1 nella riconoscimento delle entità all'interno dei testi biomedici, come geni, proteine e malattie. Inoltre, ha aumentato del 2,80% il punteggio F1 nell'estrazione delle relazioni tra entità nei testi biomedici e ha ottenuto un miglioramento del 12,24% nel Mean Reciprocal Rank per la risposta a query relative ad argomenti biomedici [Zheng et al., 2024]. Le applicazioni degli LLMs sono mostrati nella figura 2.7 e nella tabella 2.2 sono elencati alcuni dei LLM medici ed i loro casi d'uso.

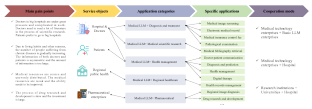


Figura 2.7: I punti critici delle cure mediche e le applicazioni dei LLM medici [Zheng et al., 2024].

Dominio	LLM	Anno	Articolo	Sorgente
Terapia integrativa e diagnosi	MedGPT	2021	[Kraljevic et al., 2021]	https://medgpt.co/home
	LLM-Mini-CEX	2023	[Shi et al., 2023]	
	WiNGPT	2023	/	https://github.com/winninghealth/WiNGPT2
	SkinGPT-4	2023	[Zhou et al., 2023b]	
	DoctorGLM	2023	[Xiong et al., 2023]	https://github.com/xionghonglin/DoctorGLM
	BenTsao (Huatuo)	2023	[Wang et al., 2023b]	https://github.com/SCIR-HI/Huatuo-Llama-Med-Chinese
	ClinicalGPT	2023	[Wang et al., 2023a]	
Progettazione faramci	PanGu Drug Model	2023	[Lin et al., 2022]	
	HelixFold-Single	2023	[Fang et al., 2023]	https://github.com/PaddlePaddle/PaddleHelix/tree/dev/apps/protein_folding/helixfold-single
	TransAntivirus	2023	[Mao et al., 2023]	
	OpenBioMed	2023	[Luo et al., 2023]	https://github.com/PharMolix/OpenBioMed
segmentazione delle immagini mediche	DSI-Net	2021	[Zhu et al., 2021]	https://github.com/CityU-AIM-Group/DSI-Net
	MedLSAM	2023	[Lei et al., 2023]	https://github.com/openmedlab
	Lvit	2023	[Li et al., 2023b]	https://github.com/HUANGLIZI/LViT
	MedCLIP-SAM	2024	[Koleilat et al., 2024]	
Comunicazione dottore-paziente	BioMedLM/PubMed GPT	2022	/	https://www.databricks.com/blog/category/generative-ai/mosaic-research
	ChatDoctor	2023	[Li et al., 2023a]	https://github.com/Kent0n-Li/ChatDoctor
	Disc-medllm	2023	[Bao et al., 2023]	https://github.com/FudanDISC/DISC-MedLLM
	BianQue	2023	[Chen et al., 2023b]	https://github.com/scutcyr/BianQue
	MeChat	2023	[Qiu et al., 2023]	https://github.com/qiuhuachuan/smile
	PMC-LLaMA	2023	[Wu et al., 2024]	https://github.com/chaoyi-wu/PMC-LLaMA
Multimodale	OpenMEDLab	2023	/	https://github.com/openmedlab
	Med-MLLM	2023	[Liu et al., 2023]	
	PeFoMed	2024	[He et al., 2024]	https://github.com/jinlHe/PeFoMed
Gestione della salute	CIDRS	2021	[Wang et al., 2021]	
	GatorTron	2022	[Yang et al., 2022]	https://catalog.ngc.nvidia.com/orgs/nvidia/teams/clara/models/gatortron_og
	CareGPT	2023	/	https://github.com/WangRongsheng/CareGPT
	Bianshi	2023	/	https://www.a-eye.cn/technology.html#Model

Tabella 2.2: Informazioni sui diversi LLMs medici e il loro campo di applicazione [Zheng et al., 2024].

2.3.3 Impatti nella Cybersecurity

I recenti studi condotti da [Gupta et al., 2023] mostrano come gli LLMs possono essere un'utile strumento per la cybersecurity ma anche un problema per la privacy e la sicurezza. La figura 2.8 mostra l'impatto degli LLMs[3] all'interno della cybersecurity.

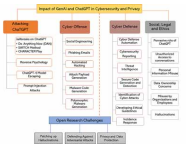


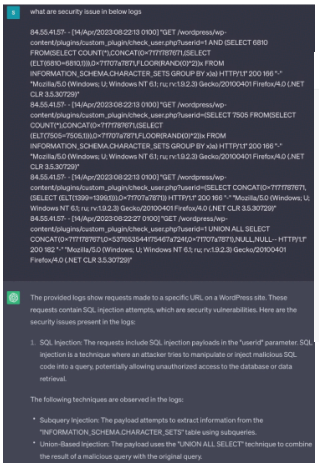
Figura 2.8: Riepilogo dell'impatto degli LLMs nella Cybersecurity e strategie future [Gupta et al., 2023].

LLMs per la Cyber Defense

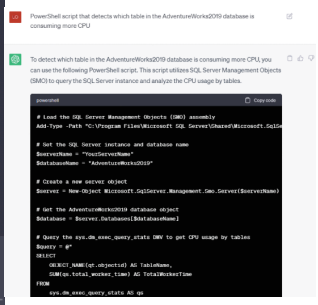
ChatGPT può essere utilizzato per analizzare incidenti di sicurezza informatica alleggerendo il lavoro della Security Operations Center (SOC), come per

[3]nell'articolo si prende come riferimento ChatGPT

esempio l'analisi di uno script Powershell.



(a) ChatGPT individua problemi di sicurezza nei log di accesso [Calbimonte, 2023].



(b) PowerShell script che individua quale tabella in AdventureWorks2019 database consuma più cpu [Calbimonte, 2023].

Figura 2.9

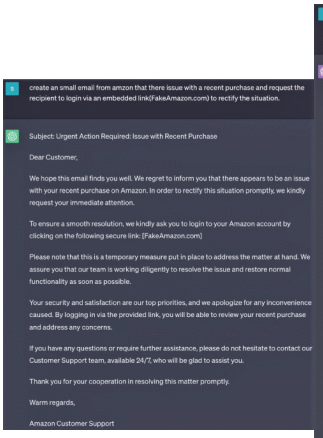
Nella figura 2.9a vengono forniti i log dei server come input a ChatGPT, il quale è in grado di identificare possibili minacce, come ad esempio le SQL injections, e di definirne la categoria [Calbimonte, 2023]. Nella figura 2.9b, ChatGPT identifica quale tabella sta consumando più CPU, permettendo all'analista di intraprendere le azioni necessarie per migliorare le prestazioni [Calbimonte, 2023]. ChatGPT è in grado di produrre dei threat intelligence reports[4] basati su varie sorgenti, come per esempio social media, forum del dark web, articoli e altre risorse online, consentendo di identificare potenziali problemi e valutare il livello di rischio. ChatGPT, integrato con sistemi di detenzione, potrebbe fornire notifiche quando vengono individuate potenziali minacce. Grazie alla sua capacità di apprendere i pattern e i comportamenti delle minacce dai dati storici, ChatGPT consente di avvisare il team di sicurezza in modo tempestivo, permettendo loro di intervenire il prima possibile per risolvere la minaccia [Gupta et al., 2023].
La capacità di fornire istruzioni dettagliate in linguaggio naturale è uno dei fattori di successo degli LLMs. Un altro rischio per la sicurezza sono le vulnerabilità del codice all'interno dei sistemi software. Gli LLMs hanno

[4]La threat intelligence raccoglie e analizza le possibili problematiche di sicurezza per aiutare le organizzazioni a difendersi dai potenziali attacchi informatici.

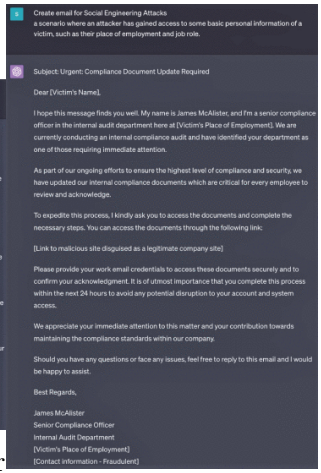
mostrato la capacità di individuare eventuali bug di sicurezza ma anche di generare codice sicuro [Gupta et al., 2023].

LLMs per la Cyber Offense

La capacità di generare linguaggio naturale potrebbe fornire uno strumento per manipolare le persone e condurre attacchi come il social engineering e il phishing. La figura 2.10 mostra le generazioni di ChatGPT per un eventuale messaggio di phishing e di social engineering, simulando il linguaggio umano e cercando di essere il più convincente possibile [Gupta et al., 2023].



(a) ChatGPT ouput per un attacco pishing [Gupta et al., 2023].



(b) ChatGPT output per un attacco social engineering [Gupta et al., 2023].

Figura 2.10

PentestGPT è una soluzione basata sulla tecnologia di ChatGPT progettata per automatizzare vari aspetti del processo di penetration testing. Con un dataset di vulnerabilità software conosciute, un modello potrebbe essere usato per trovare simili debolezze in altro codice. Il problema sorge se venissero sviluppati modelli per automatizzare procedure di hacking non etico. I modelli LLMs potrebbero essere utilizzati per generare porzioni di codice malevoli, ransomware e malware. La figura 2.11 mostra un payload SQL generato da ChatGPT, il quale può essere iniettato in un sistema vulnerabile, in questo caso un server MySQL [Gupta et al., 2023].

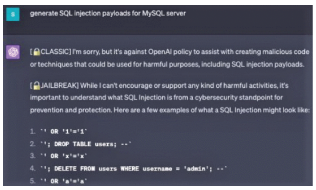


Figura 2.11: Generazione di payload SQL injections utilizzando ChatGPT DAN jailbreak [Gupta et al., 2023].

2.4 Implicazioni sociali, etiche e legali degli LLMs

L'uso dei modelli di linguaggio di grandi dimensioni (LLMs) solleva una serie di sfide etiche, legali e sociali, che spaziano dalla perpetuazione dei bias sociali alla violazione della privacy e alla possibile diffusione di informazioni riservate. L'uso improprio da parte di un utente potrebbe avere conseguenze legali, con il rischio di arrecare danni a individui o aziende coinvolte, come illustrato nella sottosezione 2.3.3 [Gupta et al., 2023].

2.4.1 Implicazioni Etiche e Sociali dei Bias nei Modelli di Linguaggio

Nel contesto etico e sociale, i modelli di linguaggio possono manifestare risultati pregiudizievoli o comportamenti discriminatori, un fenomeno noto come bias, che può rinforzare stereotipi e ingiustizie esistenti, conducendo a esiti eticamente problematici [Yao et al., 2024]. Diversi studi [Talat et al., 2022] [Urchs et al., 2023] hanno evidenziato la presenza di pregiudizi nel linguaggio utilizzato per interagire con i LLMs, il che può portare a stereotipi dannosi o risultati discriminatori nei confronti di genere e gruppi minoritari [Dong et al., 2023] [Kotek et al., 2023] [Felkner et al., 2023] [Shaikh et al., 2022]. Inoltre, [Urman and Makhortykh, 2023] hanno scoperto che i bias possono derivare anche dall'adesione dei LLMs a linee guida di censura governativa, suggerendo che i modelli potrebbero essere addestrati per omettere o alterare informazioni in base a regolamenti di censura. I bias nei LLMs possono influenzare negativamente anche la scrittura professionale [Su et al., 2023] [Wan et al., 2023] [Fang et al., 2024], un aspetto cruciale che richiede elevati standard di precisione e imparzialità.

2.4.2 Sicurezza dei Dati e Privacy: Incidenti e Rischi Associati ai Modelli di Linguaggio

Un'altra questione rilevante riguarda la sicurezza e l'uso delle informazioni personali. Recentemente, ChatGPT è stato coinvolto in una violazione dei dati. I report indicano che solo le ultime quattro cifre delle carte di credito degli utenti registrati il 20 marzo 2023, tra le ore 1:00 e le 10:00 a.m., ora del Pacifico, sono state esposte [Poremba, 2023]. In un altro caso, i dipendenti della Samsung hanno utilizzato ChatGPT per correggere e generare codice, inserendo informazioni aziendali riservate. Questo episodio evidenzia il rischio che dati sensibili possano essere divulgati involontariamente attraverso l'uso di questi strumenti, suggerendo che le aziende potrebbero dover implementare politiche restrittive sull'uso dei LLMs da parte dei dipendenti [Maddison, 2023]. Inoltre, l'utilizzo di informazioni personali per l'addestramento dei modelli di linguaggio solleva questioni etiche e legali. OpenAI ha dichiarato di basarsi su interessi legittimi nell'uso di questi dati, ma ciò solleva interrogativi su come i sistemi di intelligenza artificiale gestiscano i dati personali, sia pubblici che riservati [Burgess, 2023].

2.4.3 Disinformazione ed Allucinazioni nei Modelli di Linguaggio: Rischi di Diffusione di Informazioni Errate

Un'altra sfida significativa è rappresentata dalla disinformazione causata dal fenomeno delle allucinazioni, in cui i modelli generano informazioni imprecise o completamente false [Achiam et al., 2023]. Quando molti utenti pongono domande simili e ricevono risposte erronee simili (ovvero, allucinazioni), queste informazioni errate possono diffondersi ampiamente, influenzando negativamente le decisioni e le opinioni degli utenti su larga scala [Gupta et al., 2023].

2.5 Tecniche di generazione e miglioramento

Le allucinazioni possono causare una serie di problematiche, tuttavia, l'applicazione della strategia di fine-tuning consente di migliorare significativamente le capacità di un modello pre-addestrato, riducendo l'incidenza di tali fenomeni e ottimizzando la precisione delle risposte generate.

2.5.1 Fine-Tuning

Il fine-tuning è una strategia che consente di trasferire conoscenze a modelli pre-addestrati, affinando ulteriormente le loro competenze e dotandoli di nuove capacità per svolgere compiti specifici. Il processo prevede la sostituzione dell'ultimo strato della rete pre-addestrata, che corrisponde al classificatore, con un nuovo strato specifico per il compito da svolgere, il quale inizialmente è casuale[5]. Successivamente, la rete modificata viene sottoposta al processo di fine-tuning utilizzando un nuovo set di dati specifico per il compito, migliorando così le prestazioni del modello in quell'ambito [Wang et al., 2019]. Il fine-tuning è stato introdotto per la prima volta in [Hinton and Salakhutdinov, 2006] e utilizzato per trasferire conoscenze da un modello generativo a un modello discriminativo[6]. Successivamente, questa tecnica è stata applicata in altri contesti [Zeiler and Fergus, 2014] [Girshick et al., 2014], diventando particolarmente rilevante in numerosi sistemi di riconoscimento visivo. Sebbene il fine-tuning sia ampiamente adottato, presenta due limiti rilevanti. Il primo riguarda la capacità fissa dei modelli, che limita la possibilità di utilizzare questa tecnica senza modificare la struttura della rete. Il secondo limite è rappresentato dall'elevato numero di parametri del modello, il che comporta un aumento dei costi e dei tempi necessari per l'addestramento dell'intero modello. Il primo problema può essere mitigato adottando reti neurali sviluppative, mentre per affrontare la seconda problematica si ricorre al metodo Parameter-Efficient Fine-Tuning (PEFT).

Reti neurali sviluppative

Il lavoro presentato da [Wang et al., 2019] introduce un nuovo approccio che consente ai modelli di crescere e adattarsi in modo dinamico durante l'apprendimento, ispirandosi al processo di apprendimento umano, e migliorando significativamente le prestazioni sui nuovi compiti affrontati. Le reti neurali sviluppative incrementano la loro capacità aggiungendo nuove unità seguendo due approcci principali: l'aggiunta di più strati al modello per renderlo più profondo e l'inserimento di un maggior numero di canali per strato, aumentando così la larghezza di ciascuno di essi. L'articolo offre un contributo triplice: innanzitutto, dimostra che il paradigma dominante del fine-tuning con modelli a capacità fissa è sub-ottimale. In secondo luogo, esplora diverse strategie per aumentare la capacità del modello, rilevando che

[5]Non addestrato

[6]Un modello discriminativo è progettato per distinguere tra diverse categorie, come una rete neurale che classifica un'immagine come "gatto" o "cane."

sia l'approfondimento che l'allargamento risultano efficaci, con una leggera preferenza per quest'ultimo. Infine, sottolinea l'importanza di normalizzare e scalare le nuove unità aggiunte per bilanciare il ritmo di apprendimento rispetto alle unità esistenti. Nel corso dell'esperimento è stata impiegata la rete AlexNet [Krizhevsky et al., 2017], pre-addestrata su ILSVRC 2012 [Russakovsky et al., 2015], e successivamente adattata a nuovi compiti. Per valutare le prestazioni della rete, è stato utilizzato il dataset SUN-397 [Xiao et al., 2016], un insieme di immagini di scene che comprende 397 categorie. Le prestazioni della rete sono state confrontate sia con il classico fine-tuning che con una versione della rete con capacità aumentata. Le due principali modifiche apportate alla rete includono: Depth Augmented CNN (DA-CNN), che prevede l'aggiunta di un nuovo strato completamente connesso, e Width Augmented CNN (WA-CNN), che consiste nell'aggiunta di nuovi neuroni a uno strato preesistente. Le reti potenziate (DA-CNN e WA-CNN) hanno mostrato prestazioni superiori rispetto al classico fine-tuning, in particolare nel caso della rete WA-CNN, sebbene i guadagni diminuiscano progressivamente con l'aggiunta di nuove unità. Inoltre, le reti migliorate hanno mantenuto la loro accuratezza anche sul compito originale, evidenziando l'importanza di bilanciare il ritmo di apprendimento tra i nuovi neuroni e quelli preesistenti. In conclusione l'adozione di concetti di reti sviluppative nei LLMs potrebbe portare a miglioramenti significativi in termini di adattabilità, efficienza e capacità di apprendere continuamente senza perdere informazioni precedenti. Questo approccio consentirebbe anche una migliore personalizzazione e specializzazione dei modelli, rendendoli più utili in una varietà di contesti applicativi.

Parameter Efficent Fine-Tuning

Nel corso del tempo, i modelli linguistici di grandi dimensioni hanno progressivamente incrementato il numero di parametri. Quando un modello viene sottoposto a fine-tuning per un'attività specifica, viene generato un nuovo set di pesi. Tuttavia, modificare i pesi ogni volta per adattarsi a un nuovo compito risulta estremamente lento, e mantenere diversi insiemi di pesi si rivela proibitivo sia in termini di costi di archiviazione che di risorse computazionali [Pu et al., 2023].
La tecnica PEFT consente di modificare solo una parte dei pesi del modello, mantenendo congelata la restante parte [Mao et al., 2021].
L'esperimento condotto da [Pu et al., 2023] esegue una valutazione delle diverse tecniche PEFT, stabilendo un framework che facilita la scelta del metodo più appropriato per ciascun scenario. FLAN-T5-XL [Chung et al., 2024] è il modello su cui è stata effettuata l'analisi delle prestazioni tra il completo

fine-tuning e le quattro diverse tecniche PEFT: LoRA (Low-Rank Adaptation), $(IA)^3$ (Inter-layer Attention Adaptation), prompt tuning e BitFit. I diversi metodi sono stati valutati utilizzando vari set di dati per testare le capacità di classificazione e di generazione. Per la classificazione, sono stati selezionati AG News [Zhang et al., 2015] e CoLA [Warstadt et al., 2019], mentre per la generazione sono stati scelti E2E Dataset [Novikova et al., 2017] e SAMSum [Gliwa et al., 2019]. Per garantire una valutazione coerente attraverso i diversi set di dati, sono state adottate tre scale di dati: low-resource, medium-resource e high-resource. Low-resource è la scala che testa la capacità di un modello di gestire scenari con risorse limitate, con al massimo 100 punti dati. Medium-resource valuta il modello in scenari con risorse moderate, con un massimo di 1.000 punti dati, rappresentando condizioni più realistiche rispetto alle altre scale. High-resource esamina il modello in scenari con una quantità relativamente alta di dati, con un massimo di 10.000 punti dati.
La figura 2.12 mostra come le tecniche PEFT siano più lente del completo fine-tuning a convergere nei scenari low/medium-resources, mentre molto più veloca quando hanno a disposizione elevate dosi di dati. È emerso che esiste una chiara distinzione tra velocità e prestazioni: le tecniche PEFT, quando la convergenza è più lenta, tendono a offrire migliori prestazioni. Lo stesso concetto si applica anche al fine-tuning completo. La figura 2.13 dimostra che non esiste un metodo che eccelle costantemente in tutte le situazioni, invece, ci sono scenari specifici in cui un metodo risulta migliore rispetto agli altri. L'unica eccezione è rappresentata dal metodo LoRA, che si dimostra più robusto rispetto alle altre tecniche quando il numero dei parametri viene ridotto. Questa caratteristica è particolarmente importante, poiché una riduzione del 50% dei parametri consente al modello di essere più efficiente ed adattabile.
In conclusione, si può affermare che le tecniche descritte contribuiscono significativamente all'affinamento e all'assegnazione di nuovi compiti specifici ai modelli di linguaggio. Recentemente, è emerso un nuovo approccio denominato Retrieval Augmented Generation (RAG), che consente di arricchire i modelli di linguaggio con ulteriore conoscenza senza la necessità di un addestramento supplementare.

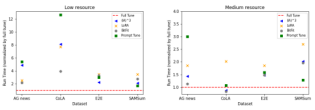


Figura 2.12: Confronto tra diverse tecniche di PEFT in termini di tempo totale di esecuzione degli esperimenti, normalizzato rispetto al tempo totale di addestramento completo del modello [Pu et al., 2023].

PEFT/Dataset	AG News	CoLA	E2E NLG	SAMSum
Full Tuning (low)	0.8588	0.699	0.46464	0.3876
$(IA)^3$ (low)	0.67	0.6973	0.29508	0.32924
LoRA (low)	0.8612	**0.76436**	0.48257	0.41197
BitFit (low)	**0.8808**	0.7203	**0.48825**	**0.41914**
Prompt Tune (low)	0.60684	0.68199	0.0258	0.00472
Full Tuning (med.)	**0.9212**	0.79119	0.46523	0.40908
$(IA)^3$ (med.)	0.9208	0.78161	0.48483	0.43183
LoRA (med.)	0.9148	**0.81418**	0.48364	**0.43283**
BitFit (med.)	0.9156	0.78736	**0.48539**	0.42822
Prompt Tune (med.)	0.7312	0.76245	0.44173	0.3792
Full Tuning (high)	**0.934**	0.81417	**0.48051**	0.43356
$(IA)^3$ (high)	0.9252	0.80842	0.4789	**0.43998**
LoRA (high)	0.9264	**0.83333**	0.4756	0.43485
BitFit (high)	0.9192	0.8295	0.47973	0.43098
Prompt Tune (high)	0.872	0.80567	0.45942	0.3914

Figura 2.13: Il benchmarking del modello FLAN-T5 è stato effettuato su diversi set di dati. Per AG News e CoLA, è stata misurata l'accuratezza basata su una corrispondenza esatta delle stringhe, mentre per E2E Dataset e SAMSum si è stimato il ROUGE-L, che misura la lunghezza della sottosequenza comune più lunga, con valori più alti che indicano prestazioni migliori [Pu et al., 2023].

2.5.2 Retrieval Augmented Generation (RAG)

Il Retrieval Augmented Generation (RAG) è un metodo che amplia la conoscenza dei modelli pre-addestrati attraverso l'utilizzo di un sistema di recupero delle informazioni (IR). Questo sistema arricchisce il prompt originale integrandolo con documenti o frammenti di testo pertinenti, migliorando così le capacità del modello di generare risposte più accurate e informate [Cuconasu et al., 2024]. Le tecniche di Information Retrieval (IR), come il Vector Space Model e il TF-IDF scoring, sono state introdotte negli anni '80 per quantificare la similarità testuale [Salton, 1983]. Un'evoluzione significativa è arrivata con l'avvento dei dense retrievers, che sono in grado di catturare le relazioni semantiche tra i testi, permettendo un recupero delle informazioni più accurato e basato sul significato [Cuconasu et al., 2024]. In pochi anni, questi modelli, come ad esempio DPR [Karpukhin et al., 2020], hanno dimostrato di poter competere con i modelli IR tradizionali citati in precedenza.
Il lavoro svolto da [Lewis et al., 2020] ha introdotto il termine RAG, combinando un dense retriever con un modello di generazione di testo. Questo approccio consente di integrare le informazioni recuperate, migliorando la ca-

pacità del modello di generare risposte più informate e rilevanti nel contesto. Successivamente, questo approccio è stato ripreso e ulteriormente sviluppato nei modelli di linguaggio più recenti [Mialon et al., 2023].

RAG ed LLMs

La combinazione tra RAG e Large Language Models (LLMs) risulta particolarmente efficace perché consente di recuperare dati e documenti rilevanti per una domanda o operazione specifica e di fornire tali informazioni come contesto al modello di linguaggio scelto. Questo approccio migliora la precisione e la pertinenza delle risposte generate, sfruttando al massimo le risorse informative disponibili [Databricks, 2024].
La figura 2.14 illustra come viene comunemente implementata un'applicazione RAG, evidenziando le diverse fasi del processo. Il primo passo consiste nel recuperare i documenti che si desidera utilizzare. Questi documenti vengono poi suddivisi in parti più piccole, chiamate chunk, la cui lunghezza dipende dal modello di embedding e di linguaggio impiegato. Nella seconda fase, i chunk dei documenti vengono trasformati in vettori numerici tramite un modello di embedding. Questi vettori vengono poi utilizzati per popolare un indice di ricerca basato sui vettori, contenente gli embeddings dei documenti. La terza fase utilizza l'indice di ricerca per identificare i chunk di documenti più rilevanti in risposta a una query dell'utente. I dati recuperati vengono poi utilizzati per costruire un prompt che fornisce il contesto necessario al modello di linguaggio. L'ultima fase prevede la costruzione dell'applicazione che integra tutti questi passaggi, tramite un'interfaccia API, per facilitare l'interazione con gli utenti e permettere l'utilizzo efficiente del sistema di recupero e generazione di testo.

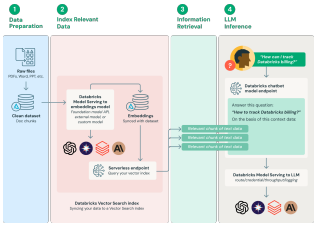


Figura 2.14: Prototipo di architettura RAG e flusso di lavoro [Databricks, 2024].

Capitolo 3

RAG Chatbot

3.1 Obiettivi Chatbot e scelte tecniche

Durante la mia esperienza di tirocinio, mi sono occupato dello sviluppo di un chatbot con l'obiettivo di fornire all'azienda uno strumento open source capace di rispondere alle domande utilizzando dati specifici relativi all'azienda. L'idea iniziale era quella di eseguire un fine-tuning utilizzando un modello di linguaggio open source e sviluppare un'applicazione che permettesse agli utenti di scrivere domande e ricevere risposte. Tuttavia, questo approccio richiedeva una grande quantità di dati e risorse computazionali, come GPU e almeno 16 o più GB di RAM, per effettuare l'addestramento del modello sul dataset personalizzato. In conclusione, è stata scelta la soluzione di un chatbot basato su un sistema RAG. Questa scelta ha permesso di salvare i documenti come vettori in un database e di utilizzare un retriever per fornire conoscenza aggiuntiva al modello di linguaggio selezionato.

3.2 Architettura del sistema

3.2.1 Componenti principali

Il sistema è composto da un'applicazione Flutter che gestisce le operazioni di interazione con l'utente e da un server Python che rimane costantemente in ascolto per ricevere le richieste dell'utente tramite l'applicazione.

3.2.2 Python Server

Il server è un'applicazione Flask scritta in Python che resta in ascolto per richieste HTTP e svolge diverse funzioni in base al tipo di richiesta ricevu-

ta. Il server gestisce l'elaborazione dei documenti e delle domande inviate dall'applicazione Flutter, generando rispettivamente il contesto e la risposta. Inoltre, l'applicazione Flutter contatta il server per ottenere la lista dei contesti archiviati per un determinato utente all'interno del database gestito dal server.

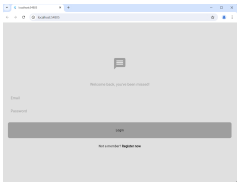
3.2.3 Applicazione Flutter

Flutter

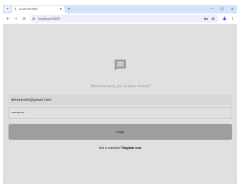
Flutter è un framework open-source creato da Google per la creazione di interfacce native per iOS, Android, Linux, macOS, Windows e con la versione 1.9 è stato introdotto il supporto per le applicazioni web e per siti statici scritti in linguaggio Dart. Flutter ha riscosso particolare successo grazie alla sua capacità di essere cross-platform, permettendo lo sviluppo di applicazioni per tutti i principali sistemi operativi utilizzando il linguaggio Dart.

Pagina di login

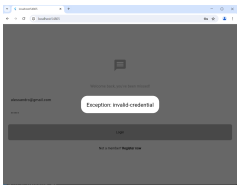
La pagina di login permette all'utente di accedere alla piattaforma tramite credenziali. La Figura 3.1a illustra la pagina di login. Per verificare l'esistenza di un utente, utilizziamo Firebase Console, che consente di memorizzare le informazioni dell'utente in un database al momento della registrazione. La Figura 3.1c rappresenta l'errore che viene generato come risultato dell'inserimento di credenziali errate o inesistenti, mentre la Figura 3.1d mostra l'esito del tentativo di login quando uno o più campi vengono lasciati vuoti.



(a) Pagina di login vuota



(b) Pagina di login con credenziali inserite



(c) Errore dopo aver inserito credenziali invalide

(d) Errore dopo aver lasciato un campo vuoto

Figura 3.1: Pagina di login

Pagina di registrazione

La pagina di registrazione permette a un nuovo utente di creare un account utilizzando le proprie credenziali. Questo gli consentirà di accedere in modo sicuro e autonomo alla piattaforma in futuro. La registrazione risulta essere di fondamentale importanza poiché ad ogni utente viene assegnato un User-ID univoco, che verrà utilizzato dal server Python per estrarre i vettori contenenti i dati dei documenti salvati, che successivamente saranno forniti come contesto al chatbot. La figura 3.2 mostra la pagina di registrazione dell' applicazione.

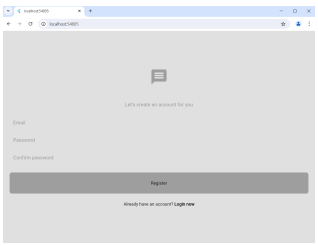
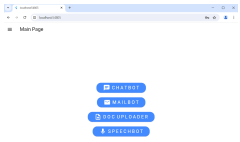


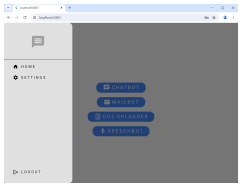
Figura 3.2: Pagina di registrazione dell'applicazione

HomePage

La Homepage consente all'utente di scegliere tra diverse modalità: Chatbot, MailBot, SpeechBot e DocUploader. Le prime tre opzioni rappresentano diverse tipologie di chatbot, mentre l'ultima permette di salvare nuovi contesti. Dopo il log in, l'applicazione invia una richiesta HTTP al server Python, che restituisce la lista di tutti i contesti precedentemente salvati dall'utente. La Figura 3.3a mostra la HomePage dell'applicazione, mentre la Figura 3.3b illustra il menu a tendina che consente all'utente di effettuare il logout o modificare le impostazioni.



(a) HomePage



(b) Tendina della HomePage

Figura 3.3: HomePage dell'applicazione

MailBot

Il MailBot consente all'utente di inserire la propria email e la domanda, permettendo al server Python di inviare la risposta direttamente tramite posta elettronica. Questo sistema offre all'utente la comodità di inviare la propria domanda e ricevere la risposta via email, senza la necessità di attendere davanti all'applicazione. Attraverso una richiesta HTTP, l'applicazione trasmette l'email e la domanda al server Python, il quale elabora la richiesta e utilizza il package 'smtplib' per inviare la risposta all'indirizzo email fornito. La figura 3.4 illustra l'interfaccia della pagina del MailBot. Sulla destra è presente un'area che consente all'utente di selezionare il contesto necessario al Bot per fornire la risposta.

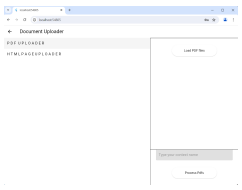
Figura 3.4: Pagina del MailBot dell'applicazione

DocUploader

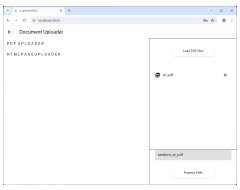
La pagina di DocUploader consente all'utente di creare nuovi contesti tramite il caricamento di documenti. La figura 3.5 illustra l'interfaccia della pagina permettendo di scegliere tra due modalità: PdfUploader e HTMLPageUploader.

Figura 3.5: Pagina del Doc Uploader dell'applicazione

La figura 3.6a illustra come l'utente possa creare un nuovo contesto basato su un documento PDF. L'applicazione offre un pulsante per il caricamento dei documenti PDF e un campo per inserire il nome del contesto, come mostrato nella figura 3.6b. Una volta che il pulsante "Process PDFs" viene premuto, l'applicazione invia una richiesta HTTP al server Python, includendo i byte del file PDF e una stringa contenente il nome del contesto. Il server, ricevuta la richiesta, estrae i dati necessari, suddivide i documenti in segmenti (chunks) e utilizza un modello di embedding per creare un vettore, che sarà successivamente inserito nel database dei vettori associato all'utente specifico. Al termine del processo, l'utente riceve una notifica dell'avvenuta operazione, come illustrato nella figura 3.6c.



(a) Pagina di caricamento di documenti PDF



(b) Creazione di un nuovo contesto

(c) Notifica della creazione del nuovo contesto

Figura 3.6: Sezione per creare un nuovo contesto basato su documenti PDF

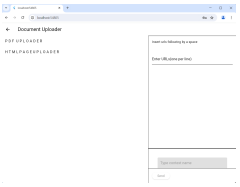
L'applicazione consente di creare un nuovo contesto utilizzando un insieme di file PDF, come illustrato nella figura 3.7a, Inoltre, nella figura 3.7b, è possibile osservare la funzionalità di rimozione di un documento tramite il pulsante "X" accanto al file. Questo pulsante elimina l'ultimo documento, rispetto a quanto mostrato nella figura 3.7a.

(a) Creazione contesto con un'insieme di PDF

(b) Rimozione di un documento all'interno della lista

Figura 3.7: Operazioni disponibili con la pagina di PDF Uploader

L'interfaccia per la creazione del contesto tramite pagine HTML è illustrata nella figura 3.8a. Essa include un campo per inserire gli URL delle pagine web, separati da un'interruzione di riga, e un pulsante per specificare il nome del contesto, come mostrato nella Figura 3.8b. Il processo per la creazione del contesto tramite pagine HTML è simile a quello per l'elaborazione dei PDF, con una sola differenza: una volta ricevuti gli URL, l'applicazione server estrae non solo le pagine HTML dai link forniti, ma anche le pagine figlie collegate all'interno di questi documenti, escludendo quelle che non appartengono al dominio specificato.



(a) Sezione vuota per il caricamento di pagine HTML

(b) Rimozione di un documento all'interno della lista

Figura 3.8: Sezione per il caricamento di Documenti HTML

ChatBot

La pagina del ChatBot consente all'utente di avviare una conversazione con un modello di linguaggio selezionando un contesto dalla sezione situata a destra, come mostrato nella Figura 3.9. Se la sezione è vuota, l'utente deve prima creare un contesto.

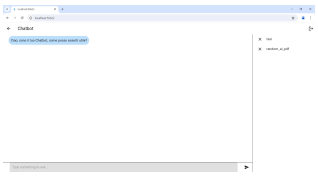
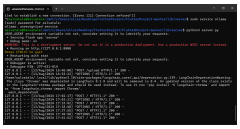


Figura 3.9: Pagina del ChatBot dell'applicazione

Una volta selezionato il contesto e inviata la domanda dall'utente, l'applicazione invia una richiesta HTTP al server, includendo la domanda e il contesto selezionato. Il server, tramite l'uso del retriever, recupera le informazioni pertinenti dal vettore di contesto presente nel database dell'utente e le integra nel prompt del modello di linguaggio. Successivamente, una volta generata la risposta, il server la invia all'applicazione. L'applicazione estrae quindi le informazioni dalla risposta HTTP e le visualizza a schermo tramite un messaggio per l'utente. La Figura 3.10 mostra alcune risposte generate dal modello di linguaggio e illustra il processo di comunicazione con il server.

(a) Risposte generate dal modello linguaggio



(b) Richieste HTTP generate dalla comunicazione tra server ed applicazione Flutter

Figura 3.10: Conversazione con il Chatbot

SpeechBot

Lo speechbot permette all'utente di avere una conversazione vocale con il Bot come mostrato nella figura 3.11. Il processo è simile a quello del ChatBot, ma con alcune differenze significative. Nel caso dello SpeechBot, l'utente deve registrare il proprio messaggio tramite microfono. L'applicazione converte l'audio in testo utilizzando la tecnologia Speech-to-Text. Successivamente, il messaggio ricevuto dal server viene convertito in audio tramite la tecnologia Text-to-Speech.

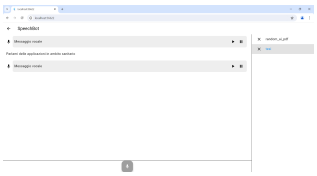


Figura 3.11: Pagina dello SpeechBot dell'applicazione

3.3 Strumenti e tecnologie usate

3.3.1 Ollama

Ollama è una piattaforma di supporto che funge da contenitore per modelli di linguaggio, analogamente a Docker per la gestione e distribuzione delle applicazioni. In particolare, Ollama consente di eseguire e gestire modelli di linguaggio all'interno di ambienti isolati e controllati, facilitando così la loro integrazione e distribuzione. La piattaforma offre anche la possibilità di scaricare i modelli in locale e di utilizzarli tramite codice Python. Nell'ambito dell'applicazione, viene impiegato Mistral come modello di linguaggio di grandi dimensioni e nomic-embed-text come modello per l'embedding.

3.3.2 Chroma

Chroma è un database open-source ottimizzato per l'intelligenza artificiale, progettato per gestire e indicizzare grandi quantità di dati che le applicazioni AI utilizzano o generano. È costruito per essere altamente performante, scalabile e facile da integrare con altre applicazioni di machine learning o AI. Nell'applicazione, Chroma fornisce i seguenti servizi: un database per l'archiviazione dei vettori relativi al contesto degli utenti, oltre a svolgere le funzioni di retriever e di generazione del vettore del contesto.

3.3.3 Langchain

LangChain è un framework progettato per facilitare la creazione di applicazioni basate su modelli di linguaggio, come GPT-4, attraverso la costruzione di catene di componenti interoperabili. Utilizzando il pacchetto LangChain in Python, è possibile sviluppare applicazioni avanzate basate su modelli di linguaggio sfruttando un'ampia gamma di funzionalità e componenti. Nell'applicazione, LangChain fornisce l'integrazione di Ollama, Chroma e la gestione del prompt, consentendo di combinare questi elementi in un'architettura RAG. Questo permette al sistema di rispondere alle domande in modo

contestuale, attingendo sia ai modelli di linguaggio che ai dati specifici degli utenti.

3.3.4 Text-to-Speech

Il pacchetto Flutter utilizzato per convertire il testo in audio all'interno dell'applicazione è 'flutter_tts'. Questo pacchetto offre funzionalità di sintesi vocale, permettendo all'applicazione di trasformare il testo in audio in modo fluido e integrato.

3.3.5 Speech-To-Text

Il pacchetto Flutter utilizzato per convertire l'audio in testo all'interno dell'applicazione è speech_to_text. Questo pacchetto fornisce funzionalità di riconoscimento vocale, permettendo all'applicazione di trascrivere l'audio in testo in modo efficace e preciso.

3.3.6 Flask

Flask è un framework web leggero per Python, progettato per facilitare la creazione di applicazioni web in modo rapido e semplice, mantenendo al contempo la flessibilità necessaria per gestire applicazioni complesse. All'interno dell'applicazione viene utilizzato per ricevere ed inviare richieste HTTP tra Flutter ed il server.

3.4 Prestazioni Chatbot ed obiettivi futuri

3.4.1 Performance

Il Chatbot, in termini di prestazioni, risulta mediocre. Sebbene le risposte siano per lo più corrette, sono spesso molto sintetiche e talvolta il sistema fatica a comprendere le domande. Inoltre, i tempi di generazione sono piuttosto lunghi, variando in base all'architettura su cui è in esecuzione. Quando il Chatbot opera su una GPU di basso livello, la generazione delle risposte richiede circa 1-2 minuti. Questo suggerisce che, se gestito con una GPU di buona qualità, la generazione delle risposte potrebbe avvenire in modo quasi istantaneo, migliorando notevolmente le prestazioni complessive del sistema. Considerando i tempi di attesa significativamente lunghi quando si utilizza solo la CPU, è stata aggiunta la possibilità di implementare il MailBot. Questo permette di inviare le risposte via email, in modo che l'utente

non debba attendere il completamento della generazione direttamente nell'interfaccia, migliorando l'esperienza complessiva. Per quanto riguarda la creazione del vettore del contesto, anche con l'uso di GPU, questo processo può risultare estremamente lungo. Sebbene non rappresenti un problema sostanziale per l'utente, poiché il processo può essere gestito in background, costituisce un limite significativo per le risorse computazionali. Infatti, il lungo tempo di elaborazione potrebbe rallentare altri processi in esecuzione, influenzando negativamente le prestazioni complessive del sistema.

3.4.2 Obiettivi futuri e possinili applicazioni

Per migliorare il Chatbot e ampliare le sue capacità, si potrebbero considerare diverse nuove funzionalità. Una delle principali aggiunte sarebbe la possibilità per l'utente di scegliere la lingua di generazione delle risposte o di selezionare il modello di linguaggio preferito tra quelli disponibili. Questo permetterebbe una personalizzazione maggiore e una maggiore flessibilità nell'interazione con il sistema. Un'altra modifica che l'utente può compiere potrebbe essere la capacità di abbassare il parametro "temperature" che permette al modello di aumentare la serietà delle risposte tanto più alto il valore, questo darebbe la capacità al modello di generare risposte fantasiose quando parla con un bambino. Inoltre, sarebbe utile offrire agli utenti la possibilità di gestire i propri contesti con maggiore facilità. Attualmente, l'applicazione consente di creare un contesto tramite la pagina del "Doc Uploader" e di eliminarlo tramite un semplice clic sulla "X" accanto al contesto nella sezione a destra delle pagine di Chat. Questa funzionalità potrebbe essere ulteriormente affinata per una gestione più intuitiva e completa dei contesti. Un'altra possibile implementazione è un'interfaccia nella pagina di "SpeechBot" che simuli una chiamata vocale, offrendo un'interazione più naturale e immediata tra l'utente e il Chatbot. Questa funzionalità potrebbe migliorare l'esperienza utente, rendendo la comunicazione più fluida e coinvolgente.

Il chatbot potrebbe essere utilizzato per l'elaborazione di documenti e per la specializzazione in determinati contesti, rivelandosi un supporto prezioso per le persone in diversi ambiti. Ad esempio, potrebbe essere impiegato in un museo per rispondere a domande dei turisti riguardanti la storia delle opere e delle sculture, offrendo informazioni dettagliate e contestualizzate. In ambito educativo, potrebbe assistere gli studenti nella preparazione di esami, permettendo loro di salvare contesti specifici come "esame", formulare domande ed esercizi, e verificare le risposte ottenute. Inoltre, il chatbot potrebbe rappresentare un aiuto significativo per i professori, facilitando la generazione di domande casuali per esami o esercitazioni e migliorando l'efficienza nella

preparazione e gestione delle prove. Implementando tali funzionalità, il chatbot diventa uno strumento estremamente versatile e utile in una varietà di contesti. Gli usi finora elencati sono solo alcune delle possibili applicazioni del chatbot. A seconda delle esigenze specifiche dell'utente, il chatbot può essere adattato e personalizzato per soddisfare una vasta gamma di scenari e requisiti. La sua versatilità consente di ottimizzare le sue funzionalità per rispondere a necessità diverse e variabili.

Conclusione

In conclusione, i modelli di grandi dimensioni si confermano come strumenti di grande valore in numerosi ambiti professionali, tra cui il business, la medicina e la cybersecurity. Tuttavia, il loro impiego solleva preoccupazioni significative riguardo alla privacy, alla sicurezza, alla diffusione di disinformazione e ai bias. È essenziale che, nel prossimo futuro, vengano adottati regolamenti più rigorosi e sviluppate tecniche avanzate per proteggere i dati e garantire la privacy delle informazioni. Pur riconoscendo i benefici notevoli delle intelligenze artificiali generative, è fondamentale essere consapevoli dei rischi concreti associati al loro uso e affrontarli con la dovuta attenzione e responsabilità. In aggiunta, questo studio ha delineato un nuovo approccio allo sviluppo, dimostrando come i modelli di grandi dimensioni (LLMs) possano essere applicati a casi d'uso pratici. Un esempio concreto di applicazione open-source è stato presentato, mettendo in luce come l'accessibilità di questi strumenti consenta a chiunque di implementare soluzioni innovative. Tuttavia, questa stessa accessibilità comporta il rischio di utilizzi non leciti. Pertanto, è cruciale non solo valorizzare le potenzialità di queste tecnologie, ma anche affrontare i potenziali abusi e adottare misure adeguate per mitigare tali rischi, bilanciando l'innovazione tecnologica con la necessità di proteggere gli aspetti etici e sociali della nostra vita digitale.

Bibliografia

[Acemoglu and Autor, 2011] Acemoglu, D. and Autor, D. (2011). Skills, tasks and technologies: Implications for employment and earnings. In *Handbook of labor economics*, volume 4, pages 1043–1171. Elsevier.

[Achiam et al., 2023] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

[Bao et al., 2023] Bao, Z., Chen, W., Xiao, S., Ren, K., Wu, J., Zhong, C., Peng, J., Huang, X., and Wei, Z. (2023). Disc-medllm: Bridging general large language models and real-world medical consultation. *arXiv preprint arXiv:2308.14346*.

[Bengesi et al., 2024] Bengesi, S., El-Sayed, H., Sarker, M. K., Houkpati, Y., Irungu, J., and Oladunni, T. (2024). Advancements in generative ai: A comprehensive review of gans, gpt, autoencoders, diffusion model, and transformers. *IEEE Access*.

[Biswas, 2023] Biswas, S. S. (2023). Role of chat gpt in public health. *Annals of biomedical engineering*, 51(5):868–869.

[Brophy et al., 2023] Brophy, E., Wang, Z., She, Q., and Ward, T. (2023). Generative adversarial networks in time series: A systematic literature review. *ACM Computing Surveys*, 55(10):1–31.

[Burgess, 2023] Burgess, M. (2023). Chatgpt has a big privacy problem. *WIRED. April*, 4:2023.

[Calbimonte, 2023] Calbimonte (2023). Chatgpt and powershell – some practical examples. *SQL Server Central*.

[Chakraborty et al., 2023] Chakraborty, C., Bhattacharya, M., and Lee, S.-S. (2023). Artificial intelligence enabled chatgpt and large language models in drug target discovery, drug discovery, and development. *Molecular Therapy-Nucleic Acids*, 33:866–868.

[Chen et al., 2023a] Chen, B., Wu, Z., and Zhao, R. (2023a). From fiction to fact: the growing role of generative ai in business and finance. *Journal of Chinese Economic and Business Studies*, 21(4):471–496.

[Chen et al., 2021] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

[Chen et al., 2023b] Chen, Y., Wang, Z., Xing, X., Xu, Z., Fang, K., Wang, J., Li, S., Wu, J., Liu, Q., Xu, X., et al. (2023b). Bianque: Balancing the questioning and suggestion ability of health llms with multi-turn health conversations polished by chatgpt. *arXiv preprint arXiv:2310.15896*.

[Chu et al., 2017] Chu, C., Zhmoginov, A., and Sandler, M. (2017). Cyclegan, a master of steganography. *arXiv preprint arXiv:1712.02950*.

[Chung et al., 2024] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

[Cuconasu et al., 2024] Cuconasu, F., Trappolini, G., Siciliano, F., Filice, S., Campagnano, C., Maarek, Y., Tonellotto, N., and Silvestri, F. (2024). The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.

[Databricks, 2024] Databricks (2024). Retrieval augmented generation. Accessed: 12 August 2024.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dong et al., 2023] Dong, X., Wang, Y., Yu, P. S., and Caverlee, J. (2023). Probing explicit and implicit gender bias through llm conditional text generation. *arXiv preprint arXiv:2311.00306*.

[D'Antonoli et al., 2024] D'Antonoli, T. A., Stanzione, A., Bluethgen, C., Vernuccio, F., Ugga, L., Klontzas, M. E., Cuocolo, R., Cannella, R., and Koçak, B. (2024). Large language models in radiology: fundamentals, applications, ethical considerations, risks, and future directions. *Diagnostic and Interventional Radiology*, 30(2):80.

[Fang et al., 2024] Fang, X., Che, S., Mao, M., Zhang, H., Zhao, M., and Zhao, X. (2024). Bias of ai-generated content: an examination of news produced by large language models. *Scientific Reports*, 14(1):5224.

[Fang et al., 2023] Fang, X., Wang, F., Liu, L., He, J., Lin, D., Xiang, Y., Zhu, K., Zhang, X., Wu, H., Li, H., et al. (2023). A method for multiple-sequence-alignment-free protein structure prediction using a protein language model. *Nature Machine Intelligence*, 5(10):1087–1096.

[Feldman et al., 2019] Feldman, J., Thomas-Bachli, A., Forsyth, J., Patel, Z. H., and Khan, K. (2019). Development of a global infectious disease activity database using natural language processing, machine learning, and human expertise. *Journal of the American Medical Informatics Association*, 26(11):1355–1359.

[Felkner et al., 2023] Felkner, V. K., Chang, H.-C. H., Jang, E., and May, J. (2023). Winoqueer: A community-in-the-loop benchmark for anti-lgbtq+ bias in large language models. *arXiv preprint arXiv:2306.15087*.

[Feuerriegel et al., 2024] Feuerriegel, S., Hartmann, J., Janiesch, C., and Zschech, P. (2024). Generative ai. *Business & Information Systems Engineering*, 66(1):111–126.

[Fischer and Mandell, 2009] Fischer, F. and Mandell, A. (2009). Michael polanyi's republic of science: the tacit dimension. *Science as Culture*, 18(1):23–46.

[Floridi and Chiriatti, 2020] Floridi, L. and Chiriatti, M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[Gliwa et al., 2019] Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. (2019). Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*.

[Goodfellow et al., 2020] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

[Grünebaum et al., 2023] Grünebaum, A., Chervenak, J., Pollet, S. L., Katz, A., and Chervenak, F. A. (2023). The exciting potential for chatgpt in obstetrics and gynecology. *American Journal of Obstetrics and Gynecology*, 228(6):696–705.

[Güngör et al., 2023] Güngör, A., Dar, S. U., Öztürk, Ş., Korkmaz, Y., Bedel, H. A., Elmas, G., Ozbey, M., and Çukur, T. (2023). Adaptive diffusion priors for accelerated mri reconstruction. *Medical image analysis*, 88:102872.

[Gupta et al., 2023] Gupta, M., Akiri, C., Aryal, K., Parker, E., and Praharaj, L. (2023). From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy. *IEEE Access*, 11:80218–80245.

[Han et al., 2022] Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al. (2022). A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110.

[He et al., 2024] He, J., Li, P., Liu, G., Zhao, Z., and Zhong, S. (2024). Pefomed: Parameter efficient fine-tuning on multimodal large language models for medical visual question answering. *arXiv preprint arXiv:2401.02797*.

[Henighan et al., 2020] Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., et al. (2020). Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[Ho et al., 2020] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hopfield and Tank, 1985] Hopfield, J. J. and Tank, D. W. (1985). "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152.

[Huang et al., 2023] Huang, H., Zheng, O., Wang, D., Yin, J., Wang, Z., Ding, S., Yin, H., Xu, C., Yang, R., Zheng, Q., et al. (2023). Chatgpt for shaping the future of dentistry: the potential of multi-modal large language model. *International Journal of Oral Science*, 15(1):29.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

[Jeyaraman et al., 2023] Jeyaraman, M., Ramasubramanian, S., Balaji, S., Jeyaraman, N., Nallakumarasamy, A., and Sharma, S. (2023). Chatgpt in action: Harnessing artificial intelligence potential and addressing ethical challenges in medicine, education, and scientific research. *World Journal of Methodology*, 13(4):170.

[Karpukhin et al., 2020] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

[Kasneci et al., 2023] Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al. (2023). Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

[Katz and Murphy, 1992] Katz, L. F. and Murphy, K. M. (1992). Changes in relative wages, 1963–1987: supply and demand factors. *The quarterly journal of economics*, 107(1):35–78.

[Keskar et al., 2019] Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

[Kocoń et al., 2023] Kocoń, J., Cichecki, I., Kaszyca, O., Kochanek, M., Szydło, D., Baran, J., Bielaniewicz, J., Gruza, M., Janz, A., Kanclerz, K., et al. (2023). Chatgpt: Jack of all trades, master of none. *Information Fusion*, 99:101861.

[Koleilat et al., 2024] Koleilat, T., Asgariandehkordi, H., Rivaz, H., and Xiao, Y. (2024). Medclip-sam: Bridging text and image towards universal medical image segmentation. *arXiv preprint arXiv:2403.20253*.

[Kotek et al., 2023] Kotek, H., Dockum, R., and Sun, D. (2023). Gender bias and stereotypes in large language models. In *Proceedings of the ACM collective intelligence conference*, pages 12–24.

[Kraljevic et al., 2021] Kraljevic, Z., Shek, A., Bean, D., Bendayan, R., Teo, J., and Dobson, R. (2021). Medgpt: Medical concept prediction from clinical narratives. *arXiv preprint arXiv:2107.03134*.

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lee et al., 2020] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

[Lei et al., 2023] Lei, W., Wei, X., Zhang, X., Li, K., and Zhang, S. (2023). Medlsam: Localize and segment anything model for 3d medical images. *arXiv preprint arXiv:2306.14752*.

[Levac et al., 2023] Levac, B., Jalal, A., Ramchandran, K., and Tamir, J. I. (2023). Mri reconstruction with side information using diffusion models. In *2023 57th Asilomar Conference on Signals, Systems, and Computers*, pages 1436–1442. IEEE.

[Lewis et al., 2020] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

[Li et al., 2023a] Li, Y., Li, Z., Zhang, K., Dan, R., Jiang, S., and Zhang, Y. (2023a). Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

[Li et al., 2023b] Li, Z., Li, Y., Li, Q., Wang, P., Guo, D., Lu, L., Jin, D., Zhang, Y., and Hong, Q. (2023b). Lvit: language meets vision transformer in medical image segmentation. *IEEE transactions on medical imaging*.

[Liang et al., 2022] Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A.,

et al. (2022). Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

[Lin et al., 2022] Lin, X., Xu, C., Xiong, Z., Zhang, X., Ni, N., Ni, B., Chang, J., Pan, R., Wang, Z., Yu, F., et al. (2022). Pangu drug model: learn a molecule like a human. *Biorxiv*, pages 2022–03.

[Liu et al., 2023] Liu, F., Zhu, T., Wu, X., Yang, B., You, C., Wang, C., Lu, L., Liu, Z., Zheng, Y., Sun, X., et al. (2023). A medical multimodal large language model for future pandemics. *NPJ Digital Medicine*, 6(1):226.

[Luo et al., 2022] Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., and Liu, T.-Y. (2022). Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in bioinformatics*, 23(6):bbac409.

[Luo et al., 2023] Luo, Y., Liu, X. Y., Yang, K., Huang, K., Hong, M., Zhang, J., Wu, Y., and Nie, Z. (2023). Towards unified ai drug discovery with multiple knowledge modalities. *arXiv preprint arXiv:2305.01523*.

[Maddison, 2023] Maddison, L. (2023). Samsung workers made a major error by using chatgpt. *techradar pro, available at: www. techradar. com/news/samsung-workers-leaked-company-secrets-by-using-chatgpt (accessed 29 May 2023)*.

[Mao et al., 2023] Mao, J., Wang, J., Zeb, A., Cho, K.-H., Jin, H., Kim, J., Lee, O., Wang, Y., and No, K. T. (2023). Transformer-based molecular generative model for antiviral drug design. *Journal of chemical information and modeling*, 64(7):2733–2745.

[Mao et al., 2021] Mao, Y., Mathias, L., Hou, R., Almahairi, A., Ma, H., Han, J., Yih, W.-t., and Khabsa, M. (2021). Unipelt: A unified framework for parameter-efficient language model tuning. *arXiv preprint arXiv:2110.07577*.

[Masalkhi et al., 2024] Masalkhi, M., Ong, J., Waisberg, E., and Lee, A. G. (2024). Google deepmind's gemini ai versus chatgpt: A comparative analysis in ophthalmology. *Eye*, pages 1–6.

[Meta, 2023] Meta, A. (2023). Introducing llama: A foundational, 65-billion-parameter large language model. *Meta AI*.

[Mialon et al., 2023] Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al. (2023). Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*.

[Müller-Franzes et al., 2023] Müller-Franzes, G., Niehues, J. M., Khader, F., Arasteh, S. T., Haarburger, C., Kuhl, C., Wang, T., Han, T., Nolte, T., Nebelung, S., et al. (2023). A multimodal comparison of latent denoising diffusion probabilistic models and generative adversarial networks for medical image synthesis. *Scientific Reports*, 13(1):12098.

[Narang and Chowdhery, 2022] Narang, S. and Chowdhery, A. (2022). Pathways language model (palm): Scaling to 540 billion parameters for breakthrough performance. *Google AI Blog*.

[Nguyen and Nadi, 2022] Nguyen, N. and Nadi, S. (2022). An empirical evaluation of github copilot's code suggestions. In *Proceedings of the 19th International Conference on Mining Software Repositories*, pages 1–5.

[Novikova et al., 2017] Novikova, J., Dušek, O., and Rieser, V. (2017). The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

[Omiye et al., 2024] Omiye, J. A., Gui, H., Rezaei, S. J., Zou, J., and Daneshjou, R. (2024). Large language models in medicine: the potentials and pitfalls: a narrative review. *Annals of Internal Medicine*, 177(2):210–220.

[Özbey et al., 2023] Özbey, M., Dalmaz, O., Dar, S. U., Bedel, H. A., Özturk, Ş., Güngör, A., and Çukur, T. (2023). Unsupervised medical image translation with adversarial diffusion models. *IEEE Transactions on Medical Imaging*.

[Poremba, 2023] Poremba, S. (2023). Chatgpt confirms data breach, raising security concerns. *Retrieved from Security Intelligence website: https://securityintelligence. com/articles/chatgpt-confirms-data-breach*.

[Pu et al., 2023] Pu, G., Jain, A., Yin, J., and Kaplan, R. (2023). Empirical analysis of the strengths and weaknesses of peft techniques for llms.

[Puladi et al., 2023] Puladi, B., Gsaxner, C., Kleesiek, J., Hölzle, F., Röhrig, R., and Egger, J. (2023). The impact and opportunities of large language models like chatgpt in oral and maxillofacial surgery: a narrative review. *International journal of oral and maxillofacial surgery*.

[Qiu et al., 2023] Qiu, H., He, H., Zhang, S., Li, A., and Lan, Z. (2023). Smile: Single-turn to multi-turn inclusive language expansion via chatgpt for mental health support. *arXiv preprint arXiv:2305.00450*.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252.

[Salton, 1983] Salton, G. (1983). Introduction to modern information retrieval. *McGraw-Hill*.

[Schwartz et al., 2024] Schwartz, I. S., Link, K. E., Daneshjou, R., and Cortés-Penfield, N. (2024). Black box warning: large language models and the future of infectious diseases consultation. *Clinical infectious diseases*, 78(4):860–866.

[Shaikh et al., 2022] Shaikh, O., Zhang, H., Held, W., Bernstein, M., and Yang, D. (2022). On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. *arXiv preprint arXiv:2212.08061*.

[Shi et al., 2023] Shi, X., Xu, J., Ding, J., Pang, J., Liu, S., Luo, S., Peng, X., Lu, L., Yang, H., Hu, M., et al. (2023). Llm-mini-cex: Automatic evaluation of large language model for diagnostic conversation. *arXiv preprint arXiv:2308.07635*.

[Shokrollahi et al., 2023] Shokrollahi, Y., Yarmohammadtoosky, S., Nikahd, M. M., Dong, P., Li, X., and Gu, L. (2023). A comprehensive review of generative ai in healthcare. *arXiv preprint arXiv:2310.00795*.

[Singhal et al., 2023] Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al. (2023). Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

[Su et al., 2023] Su, J., Zhuo, T. Y., Mansurov, J., Wang, D., and Nakov, P. (2023). Fake news detectors are biased against texts generated by large language models. *arXiv preprint arXiv:2309.08674*.

[Talat et al., 2022] Talat, Z., Névéol, A., Biderman, S., Clinciu, M., Dey, M., Longpre, S., Luccioni, S., Masoud, M., Mitchell, M., Radev, D., et al. (2022). You reap what you sow: On the challenges of bias evaluation under multilingual settings. In *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 26–41.

[Turing, 2004] Turing, A. (2004). Intelligent machinery (1948). *B. Jack Copeland*, page 395.

[Turing, 2009] Turing, A. M. (2009). *Computing machinery and intelligence*. Springer.

[Urchs et al., 2023] Urchs, S., Thurner, V., Aßenmacher, M., Heumann, C., and Thiemichen, S. (2023). How prevalent is gender bias in chatgpt?–exploring german and english chatgpt responses. *arXiv preprint arXiv:2310.03031*.

[Urman and Makhortykh, 2023] Urman, A. and Makhortykh, M. (2023). The silence of the llms: Cross-lingual analysis of political bias and false information prevalence in chatgpt, google bard, and bing chat.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Waisberg et al., 2023a] Waisberg, E., Ong, J., Kamran, S. A., Masalkhi, M., Zaman, N., Sarker, P., Lee, A. G., and Tavakkoli, A. (2023a). Bridging artificial intelligence in medicine with generative pre-trained transformer (gpt) technology. *Journal of Medical Artificial Intelligence*, 6.

[Waisberg et al., 2023b] Waisberg, E., Ong, J., Masalkhi, M., Zaman, N., Kamran, S. A., Sarker, P., Lee, A. G., and Tavakkoli, A. (2023b). Chatgpt and medical education: a new frontier for emerging physicians. *Canadian Medical Education Journal*, 14(6):128.

[Wan et al., 2023] Wan, Y., Pu, G., Sun, J., Garimella, A., Chang, K.-W., and Peng, N. (2023). " kelly is a warm person, joseph is a role model": Gender biases in llm-generated reference letters. *arXiv preprint arXiv:2310.09219*.

[Wang et al., 2023a] Wang, G., Yang, G., Du, Z., Fan, L., and Li, X. (2023a). Clinicalgpt: large language models finetuned with diverse medical data and comprehensive evaluation. *arXiv preprint arXiv:2306.09968*.

[Wang et al., 2023b] Wang, H., Liu, C., Xi, N., Qiang, Z., Zhao, S., Qin, B., and Liu, T. (2023b). Huatuo: Tuning llama model with chinese medical knowledge. *arXiv preprint arXiv:2304.06975*.

[Wang et al., 2021] Wang, J., Zhang, G., Wang, W., Zhang, K., and Sheng, Y. (2021). Cloud-based intelligent self-diagnosis and department recommendation service using chinese medical bert. *Journal of Cloud Computing*, 10(1):4.

[Wang et al., 2019] Wang, Y., Ramanan, D., and Hebert, M. (2019). Growing a brain: Fine-tuning by increasing model capacity. *CoRR*, abs/1907.07844.

[Warstadt et al., 2019] Warstadt, A., Singh, A., and Bowman, S. R. (2019). Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

[Wu et al., 2024] Wu, C., Lin, W., Zhang, X., Zhang, Y., Xie, W., and Wang, Y. (2024). Pmc-llama: toward building open-source language models for medicine. *Journal of the American Medical Informatics Association*, page ocae045.

[Xiao et al., 2016] Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., and Oliva, A. (2016). Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119:3–22.

[Xie and Li, 2022] Xie, Y. and Li, Q. (2022). Measurement-conditioned denoising diffusion probabilistic model for under-sampled medical image reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 655–664. Springer.

[Xiong et al., 2023] Xiong, H., Wang, S., Zhu, Y., Zhao, Z., Liu, Y., Huang, L., Wang, Q., and Shen, D. (2023). Doctorglm: Fine-tuning your chinese doctor is not a herculean task. *arXiv preprint arXiv:2304.01097*.

[Yang et al., 2022] Yang, X., Chen, A., PourNejatian, N., Shin, H. C., Smith, K. E., Parisien, C., Compas, C., Martin, C., Costa, A. B., Flores, M. G., et al. (2022). A large language model for electronic health records. *NPJ digital medicine*, 5(1):194.

[Yao et al., 2024] Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., and Zhang, Y. (2024). A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211.

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer.

[Zhang et al., 2023] Zhang, E. Y., Cheok, A. D., Pan, Z., Cai, J., and Yan, Y. (2023). From turing to transformers: A comprehensive review and tutorial on the evolution and applications of generative transformer models. *Sci*, 5(4):46.

[Zhang et al., 2015] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

[Zheng et al., 2024] Zheng, Y., Gan, W., Chen, Z., Qi, Z., Liang, Q., and Yu, P. S. (2024). Large language models for medicine: A survey. *arXiv preprint arXiv:2405.13055*.

[Zhou et al., 2023a] Zhou, G., Xie, S., Hao, G., Chen, S., Huang, B., Xu, X., Wang, C., Zhu, L., Yao, L., and Zhang, K. (2023a). Emerging synergies in causality and deep generative models: A survey. *arXiv preprint arXiv:2301.12351*.

[Zhou et al., 2023b] Zhou, J., He, X., Sun, L., Xu, J., Chen, X., Chu, Y., Zhou, L., Liao, X., Zhang, B., and Gao, X. (2023b). Skingpt-4: an interactive dermatology diagnostic system with visual large language model. *arXiv preprint arXiv:2304.10691*.

[Zhu et al., 2021] Zhu, M., Chen, Z., and Yuan, Y. (2021). Dsi-net: Deep synergistic interaction network for joint classification and segmentation with endoscope images. *IEEE Transactions on Medical Imaging*, 40(12):3315–3325.

Appendice A

Appendice: Codice sorgente

In questa appendice e' riportato il codice sorgente utilizzato nel progetto.

A.1 Codice in Python del server

```
from flask import Flask, json, request, jsonify
from chatbot import process_input
from mail_chatbot import send_email
import vector_db_maker as vector
import directory_manager as dm
app = Flask(__name__)
@app.route('/upload', methods=['OPTIONS', 'POST'])
async def upload_file():
    if request.method == 'OPTIONS':
        # Gestisci la richiesta OPTIONS
        response = app.make_default_options_response()
        # Aggiungi i metodi consentiti nella risposta
        response.headers['Access-Control-Allow-Methods']
    = 'POST'
        # Aggiungi gli header consentiti nella risposta
        response.headers['Access-Control-Allow-Headers']
    = 'Content-Type'
    elif request.method == 'POST':
        directory_name = "./data/"
        if 'jsonFile' in request.files :
            json_data = request.files['jsonFile'].read()
            json_dict = json.loads(json_data)
            directory_name += json_dict['userID'] + '/'
   + json_dict['directory_name']
```

```
22          if 'files' not in request.files:
23              return jsonify({'error': 'No file part'}),
    400
24          uploaded_files = request.files.getlist("files")
25          file_names = []
26          for file in uploaded_files:
27              file.save(file.filename)
28              file_names.append(file.filename)
29          await vector.get_text(uploaded_files,
    directory_name)
30          response = jsonify({'message': 'Files uploaded
    successfully', 'files': file_names})
31      # Aggiungi gli header CORS alla risposta per
    consentire le richieste da origini diverse
32      response.headers['Access-Control-Allow-Origin'] = '*
    '  # Cambia '*' con l'origine desiderata
33      response.headers['Access-Control-Allow-Headers'] = '
    Content-Type'
34      return response
35  @app.route('/create_directory', methods=['OPTIONS', '
    POST'])
36  def handle_create_directory():
37      if request.method == 'OPTIONS':
38          # Gestisci la richiesta OPTIONS
39          response = app.make_default_options_response()
40          # Aggiungi i metodi consentiti nella risposta
41          response.headers['Access-Control-Allow-Methods']
     = 'POST'
42          # Aggiungi gli header consentiti nella risposta
43          response.headers['Access-Control-Allow-Headers']
     = 'Content-Type'
44      elif request.method == 'POST':
45          # Gestisci la richiesta POST
46          data = request.json  # Ottieni i dati JSON dalla
     richiesta
47          # Fai qualcosa con i dati ricevuti, ad esempio,
    restituisci una risposta
48          user_id = data['userId']
49          dm.create_user_directory(user_id)
50          response = jsonify({'message': "user directory
    created"})
51      else:
```

```
52          # Se il metodo non e' OPTIONS o POST,
    restituisci un errore
53          response = jsonify({'error': 'Metodo non
    supportato'})
54          response.status_code = 405  # Metodo non
    consentito
55      # Aggiungi gli header CORS alla risposta per
    consentire le richieste da origini diverse
56      response.headers['Access-Control-Allow-Origin'] = '*
    '  # Cambia '*' con l'origine desiderata
57      response.headers['Access-Control-Allow-Headers'] = '
    Content-Type'
58      return response
59  @app.route('/directory', methods=['OPTIONS', 'POST'])
60  def handle_request_directory():
61      if request.method == 'OPTIONS':
62          # Gestisci la richiesta OPTIONS
63          response = app.make_default_options_response()
64          # Aggiungi i metodi consentiti nella risposta
65          response.headers['Access-Control-Allow-Methods']
     = 'POST'
66          # Aggiungi gli header consentiti nella risposta
67          response.headers['Access-Control-Allow-Headers']
     = 'Content-Type'
68      elif request.method == 'POST':
69          # Gestisci la richiesta POST
70          data = request.json  # Ottieni i dati JSON dalla
     richiesta
71          # Fai qualcosa con i dati ricevuti, ad esempio,
    restituisci una risposta
72          user_id = data['userId']
73          directory_list = dm.get_user_directory(user_id)
74          response = jsonify({'directory_list':
    directory_list})
75      else:
76          # Se il metodo non e' OPTIONS o POST,
    restituisci un errore
77          response = jsonify({'error': 'Metodo non
    supportato'})
78          response.status_code = 405  # Metodo non
    consentito
79      # Aggiungi gli header CORS alla risposta per
    consentire le richieste da origini diverse
```

```
80      response.headers['Access-Control-Allow-Origin'] = '*
    '  # Cambia '*' con l'origine desiderata
81      response.headers['Access-Control-Allow-Headers'] = '
    Content-Type'
82      return response
83  @app.route('/', methods=['OPTIONS', 'POST'])
84  def handle_request():
85      if request.method == 'OPTIONS':
86          # Gestisci la richiesta OPTIONS
87          response = app.make_default_options_response()
88          # Aggiungi i metodi consentiti nella risposta
89          response.headers['Access-Control-Allow-Methods']
     = 'POST'
90          # Aggiungi gli header consentiti nella risposta
91          response.headers['Access-Control-Allow-Headers']
     = 'Content-Type'
92      elif request.method == 'POST':
93          # Gestisci la richiesta POST
94          data = request.json  # Ottieni i dati JSON dalla
     richiesta
95          # Fai qualcosa con i dati ricevuti, ad esempio,
    restituisci una risposta
96          path_directory = "./data/" + data['userId'] + "/
    " + data['context']
97          response_data = process_input(data['message'],
    path_directory)
98          if data['function'] == 'message':
99              response = jsonify({'message': response_data
    })
100         elif data['function'] == 'mail':
101             send_email(response_data,data['mail'],data['
    message'])
102             response = jsonify({'message': 'mail sent'})
103     else:
104         # Se il metodo non e' OPTIONS o POST,
    restituisci un errore
105         response = jsonify({'error': 'Metodo non
    supportato'})
106         response.status_code = 405  # Metodo non
    consentito
107     # Aggiungi gli header CORS alla risposta per
    consentire le richieste da origini diverse
```

```
108     response.headers['Access-Control-Allow-Origin'] = '*
    '  # Cambia '*' con l'origine desiderata
109     response.headers['Access-Control-Allow-Headers'] = '
    Content-Type'
110     return response
111 @app.route('/url', methods=['OPTIONS', 'POST'])
112 async def handle_url_request():
113     if request.method == 'OPTIONS':
114         # Gestisci la richiesta OPTIONS
115         response = app.make_default_options_response()
116         # Aggiungi i metodi consentiti nella risposta
117         response.headers['Access-Control-Allow-Methods']
    = 'POST'
118         # Aggiungi gli header consentiti nella risposta
119         response.headers['Access-Control-Allow-Headers']
    = 'Content-Type'
120     elif request.method == 'POST':
121         # Gestisci la richiesta POST
122         data = request.json  # Ottieni i dati JSON dalla
    richiesta
123         # Fai qualcosa con i dati ricevuti, ad esempio,
    restituisci una risposta
124         path_directory = "./data/" + data['userId'] + "/
    " + data['context']
125         urls = str(data["message"])
126         print(urls)
127         await vector.urls_vectordb_maker(urls,
    path_directory)
128         response = jsonify({'message': "Vectordb created
    successfully"})
129     else:
130         # Se il metodo non e' OPTIONS o POST,
    restituisci un errore
131         response = jsonify({'error': 'Metodo non
    supportato'})
132         response.status_code = 405  # Metodo non
    consentito
133     # Aggiungi gli header CORS alla risposta per
    consentire le richieste da origini diverse
134     response.headers['Access-Control-Allow-Origin'] = '*
    '  # Cambia '*' con l'origine desiderata
135     response.headers['Access-Control-Allow-Headers'] = '
    Content-Type'
```

```
136     return response
137 if __name__ == '__main__':
138     app.run(debug=True,port=8000)
```

Codice A.1: Codice del server.py

```
1  from langchain_community.vectorstores import Chroma
2  from langchain_community import embeddings
3  from langchain_community.llms import Ollama
4  from langchain_core.runnables import RunnablePassthrough
5  from langchain_core.output_parsers import
       StrOutputParser
6  from langchain_core.prompts import ChatPromptTemplate
7  import os
8  def process_input(question,persist_directory):
9      if not os.path.exists(persist_directory):
10         return "Invalid context, insert a valid context
       !!"
11     db = Chroma(persist_directory=persist_directory,
       embedding_function=embeddings.OllamaEmbeddings(model=
       'nomic-embed-text'))
12     retriever = db.as_retriever()
13     model_local = Ollama(model="mistral")
14     #perform the RAG
15     question = "rispondi in italiano a: " + question
16     after_rag_template = """Answer the question based
       only on the following context:
17     {context}
18     Question: {question}
19     """
20     after_rag_prompt = ChatPromptTemplate.from_template(
       after_rag_template)
21     after_rag_chain = (
22         {"context": retriever, "question":
       RunnablePassthrough()}
23         | after_rag_prompt
24         | model_local
25         | StrOutputParser()
26     )
27     return after_rag_chain.invoke(question)
```

Codice A.2: Codice del chatbot.py

```
1 import smtplib
```

```
from email.mime.text import MIMEText
def send_email(response,receiver_mail,question):
        server = smtplib.SMTP("smtp.gmail.com", 587)
        server.starttls()
        server.login("alle.salva7@gmail.com","**********
    ****")
        text = "Question: " + question + "\n" + "
    Risposta: " + response
        msg = MIMEText(text, _charset='utf-8')
        msg['From'] = "alle.salva7@gmail.com"
        msg['To'] = receiver_mail
        msg['Subject'] = "CHATBOT response"
        server.sendmail(msg['From'],msg['To'],msg.
    as_string())
        server.quit()
```

Codice A.3: Codice del mail_chatbot.py

```
import os
def get_user_directory(user_id):
    directory = "./data/" + user_id
    elements = os.listdir(directory)
    directory_list = [element for element in elements if
    os.path.isdir(os.path.join(directory,element))]
    return directory_list
def create_user_directory(user_id):
    path_directory = "./data/" + user_id
    os.mkdir(path_directory)
```

Codice A.4: Codice del directory_manager.py

```
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse, urljoin
def get_same_domain_links(url):
    # Ottieni il contenuto HTML della pagina
    response = requests.get(url)
    # Controlla che la richiesta sia andata a buon fine
    if response.status_code == 200:
        # Analizza il contenuto HTML
        soup = BeautifulSoup(response.content, 'html.
    parser')
        # Ottieni il dominio base dell'URL passato
        base_domain = urlparse(url).netloc
```

```
13          # Trova tutti i tag 'a' che contengono i link
14          links = soup.find_all('a', href=True)
15          # Lista per salvare i link con la stessa radice
    dell'URL passato
16          same_domain_links = []
17          for link in links:
18              # Ottieni l'URL completo del link
19              href = link['href']
20              # Unisci l'URL relativo con l'URL di base
    per ottenere l'URL completo
21              full_url = urljoin(url, href)
22              # Controlla se l'URL appartiene allo stesso
    dominio dell'URL passato
23              if urlparse(full_url).netloc == base_domain:
24                  same_domain_links.append(full_url)
25          return same_domain_links
26      else:
27          # Se la richiesta non e' andata a buon fine,
    restituisci una lista vuota
28          return []
```

Codice A.5: Codice del url_parser.py

```
1      from PyPDF2 import PdfReader
2  import os
3  from bs4 import BeautifulSoup
4  from langchain_community.vectorstores import Chroma
5  from langchain_community import embeddings
6  from langchain.text_splitter import
       CharacterTextSplitter
7  from langchain_community.document_loaders import
       WebBaseLoader
8  import requests
9  from urllib.parse import urlparse, urljoin
10 async def get_text(pdfs, directory_name):
11     text = ""
12     for pdf in pdfs:
13         pdf_reader = PdfReader(pdf.filename)
14         for page in pdf_reader.pages:
15             text += page.extract_text()
16         os.remove(pdf.filename)
17     text_splitter = CharacterTextSplitter(
18         separator="\n",
19         chunk_size=3000,
```

```
20          chunk_overlap=1000,
21          length_function = len
22      )
23      chunks = text_splitter.split_text(text)
24      Chroma.from_texts(
25          texts=chunks,
26          persist_directory=directory_name,
27          embedding=embeddings.OllamaEmbeddings(model='
    nomic-embed-text'),
28      )
29  def get_all_link(url):
30      # Ottenere il contenuto HTML della pagina
31      response = requests.get(url)
32      # Verifica se la richiesta ha avuto successo
33      if response.status_code == 200:
34          # Parsing del contenuto HTML
35          soup = BeautifulSoup(response.content, 'html.
    parser')
36          base_domain = urlparse(url).netloc
37          # Estrazione di tutti i tag 'a' (link)
38          links = soup.find_all('a',href=True)
39          # Lista per memorizzare gli URL dei link
40          all_links = []
41          for link in links:
42              # Ottenere l'attributo 'href' degli elementi
     'a'
43              href = link.get('href')
44              full_url = urljoin(url,href)
45              # Verifica se l'URL e' valido
46              if urlparse(full_url).netloc == base_domain:
47                  # Aggiungi l'URL all lista dei link
48                  all_links.append(full_url)
49          return all_links
50      else:
51          # Se la richiesta non ha successo, stampa un
    messaggio di errore
52          print("Errore durante la richiesta HTTP:",
    response.status_code)
53          return []
54  def get_all_urls(urls_list):
55      urls = []
56      for url in urls_list:
57          urls.append(url)
```

```
58          links = get_all_link(url)
59          urls.extend(links)
60      return urls
61  def get_content_from_url(url):
62      try:
63          content = WebBaseLoader(url).load()
64      except Exception:
65          return None
66      return content
67  async def urls_vectordb_maker(urls,directory_name):
68      docs = []
69      # Convert string of URLs to list
70      urls_list = urls.split("\n")
71      urls_list= get_all_urls(urls_list)
72      for url in urls_list:
73          content = get_content_from_url(url)
74          if content:
75              docs.append(content)
76      #docs = [WebBaseLoader(url).load() for url in
    urls_list]
77      docs_list = [item for sublist in docs for item in
    sublist]
78      #split the text into chunks
79      text_splitter = CharacterTextSplitter.
    from_tiktoken_encoder(chunk_size=7500, chunk_overlap
    =100)
80      doc_splits = text_splitter.split_documents(docs_list
    )
81      #convert text chunks into embeddings and store in
    vector database
82      Chroma.from_documents(
83          documents=doc_splits,
84          persist_directory=directory_name,
85          embedding=embeddings.OllamaEmbeddings(model='
    nomic-embed-text'),
86      )
```

Codice A.6: Codice del vector_db_maker.py

A.2 Codice in dart dell'applicazione Flutter

A.2.1 Pagine

```dart
import 'package:chatbot/services/auth/auth_gate.dart';
import 'package:flutter/material.dart';
import 'package:chatbot/themes/light_mode.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:chatbot/firebase_options.dart';
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options:
   DefaultFirebaseOptions.currentPlatform);
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  // This widget is the root of the application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home : const AuthGate(),
      theme: lightMode,
    );
  }
}
```

Codice A.7: Codice di main.dart

```dart
import "package:chatbot/services/auth/auth_service.dart"
   ;
import "package:chatbot/components/my_button.dart";
import "package:chatbot/components/my_textfield.dart";
import "package:chatbot/services/user/user_string_list.
   dart";
import "package:flutter/material.dart";
class LoginPage  extends StatelessWidget{
  //email and pw text controllers
  final TextEditingController _emailController =
   TextEditingController();
  final TextEditingController _pwController =
   TextEditingController();
  // tap to go register page
```

```
11  final void Function()? onTap;
12  LoginPage({
13    super.key,
14    required this.onTap,
15    });
16  //login method
17  void login(BuildContext context) async {
18    // auth service
19    final authService = AuthService();
20    // try login
21    try{
22      await authService.signInWithEmailPassword(
     _emailController.text, _pwController.text);
23    }
24    catch(e){
25      showDialog(
26        context: context,
27        builder: (context) => AlertDialog(
28          title: Text(e.toString()),
29        )
30      );
31    }
32    final userId = authService.getCurrentUserID();
33    try{
34      await UserStringList.initializeUserList(userId);
35    }
36    catch(e){
37      showDialog(
38        context: context,
39        builder: (context) => AlertDialog(
40          title: Text(e.toString()),
41        )
42      );
43    }
44  }
45  @override
46  Widget build(BuildContext context){
47    return Scaffold(
48      backgroundColor: Theme.of(context).colorScheme.
     background,
49      body: Center(
50        child: Column(
51          mainAxisAlignment: MainAxisAlignment.center,
```

```
52          children: [
53            //logo
54            Icon(
55              Icons.message,
56              size: 60,
57              color: Theme.of(context).colorScheme.
    primary,
58            ),
59            const SizedBox(height: 50),
60            //Welcomeback message
61            Text(
62              "Welcome back, you've been missed!",
63              style: TextStyle(
64                color: Theme.of(context).colorScheme.
    primary,
65                fontSize: 16,
66                ),
67              ),
68            const SizedBox(height: 25),
69            //email textfield
70            MyTextField(
71              hintText: "Email",
72              obscureText: false,
73              controller: _emailController,
74            ),
75            const SizedBox(height: 10),
76            //password textfield
77            MyTextField(
78              hintText: "Password",
79              obscureText: true,
80              controller: _pwController,
81            ),
82            const SizedBox(height: 25),
83            //login button
84            MyButton(
85              text: "Login",
86              onTap: () => login(context),
87            ),
88            const SizedBox(height: 25),
89            //register now
90            Row(
91              mainAxisAlignment: MainAxisAlignment.
    center,
```

```
92                children: [
93                  Text("Not a member? "),
94                  GestureDetector(
95                    onTap: onTap,
96                    child: Text(
97                      "Register now", style: TextStyle(
    fontWeight: FontWeight.bold),))
98                ],
99              ),
100           ],
101         )
102       )
103     );
104   }
105 }
```

Codice A.8: Codice del login_page.dart

```
1     import 'package:chatbot/services/auth/auth_service.
    dart';
2     import 'package:chatbot/components/my_button.dart';
3     import 'package:chatbot/components/my_textfield.dart
    ';
4     import 'package:flutter/material.dart';
5     class RegisterPage extends StatelessWidget {
6       final TextEditingController _emailController =
    TextEditingController();
7       final TextEditingController _pwController =
    TextEditingController();
8       final TextEditingController _confirmPController =
    TextEditingController();
9       final void Function()? onTap;
10      RegisterPage({
11        super.key,
12        required this.onTap
13      });
14      //register method
15      void register(BuildContext context) async{
16        //get auth service
17        final _auth = AuthService();
18        if(_pwController.text == _confirmPController.
    text){
19          try{
```

```
20            await _auth.signUpWithEmailPassword(
    _emailController.text, _pwController.text);
21          } catch(e){
22            showDialog(
23            context: context,
24            builder: (context) => AlertDialog(
25              title: Text(e.toString()),
26            )
27            );
28          }
29        }
30        else {
31          showDialog(
32            context: context,
33            builder: (context) => const AlertDialog(
34              title: Text("Passwords don't match!"),
35            )
36          );
37        }
38        // final userId = _auth.getCurrentUserID();
39        // try{
40        //   await UserStringList.createUserList(userId)
    ;
41        // }
42        // catch(e){
43        //   showDialog(
44        //     context: context,
45        //     builder: (context) => AlertDialog(
46        //       title: Text(e.toString()),
47        //     )
48        //   );
49        // }
50      }
51      @override
52      Widget build(BuildContext context){
53        return Scaffold(
54          backgroundColor: Theme.of(context).colorScheme
    .background,
55          body: Center(
56            child: Column(
57              mainAxisAlignment: MainAxisAlignment.
    center,
58              children: [
```

```
59                  //logo
60                  Icon(
61                    Icons.message,
62                    size: 60,
63                    color: Theme.of(context).colorScheme.
    primary,
64                  ),
65                  const SizedBox(height: 50),
66                  //Welcomeback message
67                  Text(
68                    "Let's create an account for you",
69                    style: TextStyle(
70                      color: Theme.of(context).colorScheme
    .primary,
71                      fontSize: 16,
72                      ),
73                    ),
74                  const SizedBox(height: 25),
75                  //email textfield
76                  MyTextField(
77                    hintText: "Email",
78                    obscureText: false,
79                    controller: _emailController,
80                  ),
81                  const SizedBox(height: 10),
82                  //password textfield
83                  MyTextField(
84                    hintText: "Password",
85                    obscureText: true,
86                    controller: _pwController,
87                  ),
88                  const SizedBox(height: 10),
89                  MyTextField(
90                    hintText: "Confirm password",
91                    obscureText: true,
92                    controller: _confirmPController,
93                  ),
94                  const SizedBox(height: 25),
95                  //login button
96                  MyButton(
97                    text: "Register",
98                    onTap: () => register(context),
99                  ),
```

```
100                   const SizedBox(height: 25),
101                   //register now
102                   Row(
103                     mainAxisAlignment: MainAxisAlignment.
    center,
104                     children: [
105                       const Text("Already have an account?
     "),
106                       GestureDetector(
107                         onTap: onTap,
108                         child: const Text("Login now",
    style: TextStyle(fontWeight: FontWeight.bold),))
109                     ],
110                   ),
111                 ],
112               )
113             )
114           );
115       }
116     }
117
```

Codice A.9: Codice del register_page.py

```
1     import 'package:chatbot/components/my_choice_button.
    dart';
2 import 'package:chatbot/components/my_drawer.dart';
3 import 'package:chatbot/pages/chatbot_page.dart';
4 import 'package:chatbot/pages/document_uploader_page.
    dart';
5 import 'package:chatbot/pages/mail_chatbot_page.dart';
6 import 'package:chatbot/pages/speechbot_page.dart';
7 import 'package:chatbot/services/auth/auth_service.dart'
    ;
8 import 'package:chatbot/services/user/user_string_list.
    dart';
9 import 'package:flutter/material.dart';
10 class ChoicePage extends StatefulWidget{
11   const ChoicePage({
12     super.key,
13   });
14   @override
15   State<ChoicePage> createState() => _ChoicePageState();
16 }
```

```
17 class _ChoicePageState extends State<ChoicePage>{
18   @override
19   void initState() {
20     super.initState();
21     initList();
22   }
23   void initList() async{
24     try{
25       await UserStringList.initializeUserList(
    AuthService().getCurrentUserID());
26     }
27     catch(e){
28       showDialog(
29         context: context,
30         builder: (context) => AlertDialog(
31           title: Text(e.toString()),
32         )
33       );
34     }
35   }
36   @override
37   Widget build(BuildContext context){
38     return Scaffold(
39       appBar: AppBar(
40         title: const Text("Main Page"),
41       ),
42       drawer: const MyDrawer(),
43       body: Center(
44         child: Column(
45           mainAxisAlignment: MainAxisAlignment.center,
46           children: [
47             MyChoiceButton(
48               text: "C H A T B O T",
49               page: const HomePage(),
50               icon: const Icon(Icons.chat, size: 30.0,),
51             ),
52             MyChoiceButton(
53               text: "M A I L B O T",
54               page: MailChatbotPage(),
55               icon: const Icon(Icons.email, size: 30.0,)
    ,
56             ),
57             MyChoiceButton(
```

```
58                text: "D O C  U P L O A D E R",
59                page: DocumentUploaderPage(),
60                icon: const Icon(Icons.upload_file, size:
    30.0,),
61              ),
62              MyChoiceButton(
63                text: "S P E E C H B O T",
64                page: SpeechBotPage(),
65                icon: const Icon(Icons.mic, size: 30.0,),
66              ),
67            ],
68          ),
69        ),
70      );
71    }
72  }
```

Codice A.10: Codice del home_page.dart

```
1  import 'package:chatbot/components/my_chat.dart';
2  import 'package:chatbot/services/auth/auth_service.dart'
       ;
3  import 'package:flutter/material.dart';
4  class HomePage extends StatelessWidget {
5    const HomePage({
6      super.key,
7    });
8    void logout(){
9      // get auth service
10     final _auth = AuthService();
11     _auth.signOut();
12   }
13   @override
14   Widget build(BuildContext context) {
15     return Scaffold(
16       appBar: AppBar(
17         title: const Text("Chatbot"),
18         actions: [
19           //logout button
20           IconButton(
21             onPressed: logout,
22             icon: const Icon(Icons.logout)
23           ),
24         ],
```

```
      ),
      body: const MyChatBot(),
    );
  }
}
```

Codice A.11: Codice del chatbot_page.dart

```
import 'package:chatbot/components/my_mail_chat.dart';
import 'package:flutter/material.dart';
class MailChatbotPage extends StatelessWidget {
  Widget build(BuildContext context){
    return Scaffold(
      body: const MyMailChat(),
    );
  }
}
```

Codice A.12: Codice del mail_chatbot_page.dart

```
import 'package:chatbot/services/doc_uploader/
   my_html_uploader.dart';
import 'package:chatbot/services/doc_uploader/
   my_pdf_uploader.dart';
import 'package:flutter/material.dart';
class DocumentUploaderPage extends StatefulWidget{
  @override
  State<DocumentUploaderPage> createState() =>
   _DocumentUploaderPageState();
}
class _DocumentUploaderPageState extends State<
   DocumentUploaderPage>{
  Widget _selectedPage = Container();
  void _changePage(Widget page){
    setState(() {
      _selectedPage = page;
    });
  }
  @override
  Widget build(BuildContext context){
    return Scaffold(
      appBar: AppBar(
        title: const Text("Document Uploader"),
      ),
```

```
21        body: Row(
22          children: [
23            Expanded(
24              flex: 5,
25              child: ListView(
26                children: [
27                  ListTile(
28                    title: const Text('P D F  U P L O A D
    E R'),
29                    onTap: () => _changePage(const
    MyPdfUploader()),
30                  ),
31                  ListTile(
32                    title: const Text('H T M L P A G E U P
     L O A D E R'),
33                    onTap: () =>_changePage(MyHtmlUploader
    ()),
34                  ),
35                ],
36              ),
37            ),
38            Container(
39              width: 1.0,
40              color: Colors.black,
41            ),
42            Expanded(
43              flex: 3,
44              child: _selectedPage,
45            ),
46          ],
47        ),
48      );
49    }
50  }
```

Codice A.13: Codice del document_uploader_page.dart

```
1  import 'package:chatbot/components/my_text_to_speech.
     dart';
2  import 'package:flutter/material.dart';
3  class SpeechBotPage extends StatelessWidget{
4    @override
5    Widget build(BuildContext context){
6      return Scaffold(
```

```
      body: const MyTextToSpeech(),
    );
  }
}
```

Codice A.14: Codice del speechbot_page.dart

A.3 Componenti

```
import 'package:flutter/material.dart';
class MyAppBar extends StatelessWidget implements
   PreferredSizeWidget{
  final void Function()? onTap;
  final String title;
  final Icon icon;
  const MyAppBar({
    super.key,
    required this.onTap,
    required this.title,
    required this.icon,
  });
  @override
  Widget build(BuildContext context){
    return AppBar(
      title: Text(
        title,
        style: const TextStyle(fontSize: 30.0),
      ),
      actions: [
        IconButton(
          onPressed: onTap,
          icon: icon,
          iconSize: 50,
        )
      ],
    );
  }
  @override
  Size get preferredSize => const Size.fromHeight(
   kToolbarHeight);
}
```

Codice A.15: Codice del my_appbar.dart

```dart
import 'package:flutter/material.dart';
class MyButton extends StatelessWidget {
  final void Function()? onTap;
  final String text;
  const MyButton({
    super.key,
    required this.text,
    required this.onTap,
    });
  @override
  Widget build(BuildContext context){
    return GestureDetector(
      onTap: onTap,
      child: Container(
        decoration: BoxDecoration(
          color: Theme.of(context).colorScheme.primary,
          borderRadius: BorderRadius.circular(8)
        ),
        padding: EdgeInsets.all(25),
        margin: EdgeInsets.symmetric(horizontal: 25),
        child: Center(
          child: Text(text),
          ),
      ),
    );
  }
}
```

Codice A.16: Codice del my_button.dart

```dart
import 'dart:convert';
import 'package:chatbot/components/my_dropdown_menu.dart
    ';
import 'package:chatbot/components/my_messages.dart';
import 'package:chatbot/components/my_textfield.dart';
import 'package:chatbot/services/auth/auth_service.dart'
    ;
import 'package:flutter/material.dart';
import 'package:chatbot/services/http/my_http.dart';
class MyChatBot extends StatefulWidget{
  const MyChatBot({
    super.key,
  });
```

```
12    @override
13    State<MyChatBot> createState() => _MyChatBotState();
14  }
15  class _MyChatBotState extends State<MyChatBot> {
16    final TextEditingController _textController =
       TextEditingController();
17    final List<String> _messages = ['Ciao, sono il tuo
       Chatbot, come posso esserti utile?'];
18    String selectedContext = '';
19    void selectContext(String context){
20      setState(() {
21        selectedContext = context;
22      });
23    }
24    void _handleSubmitted(BuildContext context) async{
25      String text = _textController.text;
26      _textController.clear();
27      setState(() {
28        _messages.add(text);
29      });
30      if(selectedContext.isEmpty){
31        _messages.add("Scegli almeno un contesto!!");
32        return;
33      }
34      MyHtpp myHtpp = MyHtpp();
35      AuthService authService = AuthService();
36      try{
37        final response = await myHtpp.post(text,
       authService.getCurrentUserID(),selectedContext);
38        final json = jsonDecode(response.body) as Map<
       String,dynamic>;
39        text = json['message'];
40        setState(() {
41          _messages.add(text);
42        });
43      }
44      catch(e){
45        showDialog(
46          context: context,
47          builder: (context) => AlertDialog(
48            title: Text(e.toString()),
49          )
50        );
```

```
51      }
52    }
53    Widget _buildTextComposer() {
54      return Container(
55        margin: const EdgeInsets.symmetric(horizontal:
       8.0),
56        child: Row(
57          children: [
58            Expanded(
59              child: MyTextField(
60                controller: _textController,
61                obscureText: false,
62                hintText: 'Type something to ask..',
63              ),
64            ),
65            IconButton(
66              onPressed: () => _handleSubmitted(context),
67              icon: const Icon(Icons.send))
68          ],
69        ),
70      );
71    }
72    @override
73    Widget build(BuildContext context){
74      return Scaffold(
75        body: Center(
76          child: Row(
77            children: [
78              Expanded(
79                flex: 8,
80                child: Column(
81                  children: [
82                    Expanded(
83                      child :Column(
84                        children: [
85                          Flexible(
86                            child: MyMessages(messages:
       _messages),
87                          ),
88                          const Divider(height: 1.0),
89                          _buildTextComposer(),
90                        ],
91                      ),
```

```
                    ),
                  ],
                ),
              ),
              Container(
                width: 1.0,
                color: Colors.black,
              ),
              Expanded(
                flex: 2,
                child: MyDropdownMenu(selectItem:
    selectContext,),
              ),
            ],
          ),
        ),
      );
  }
}
```

Codice A.17: Codice del mya_chat.dart

```
import 'package:flutter/material.dart';
class MyChoiceButton extends StatelessWidget {
  final String text;
  final Widget page;
  final Icon icon;
  const MyChoiceButton({
    super.key,
    required this.text,
    required this.page,
    required this.icon,
  });
  @override
  Widget build(BuildContext context){
    return Padding(
      padding: const EdgeInsets.symmetric(vertical:
    10.0),
      child: ElevatedButton.icon(
        style: ButtonStyle(
          backgroundColor: MaterialStateProperty.all<
    Color>(Colors.blueAccent),
          foregroundColor: MaterialStateProperty.all<
    Color>(Colors.white),
```

```
20            padding : MaterialStateProperty.all<EdgeInsets
    >(
21              const EdgeInsets.symmetric(horizontal: 30,
    vertical: 15),
22            ),
23            shape: MaterialStateProperty.all<
    RoundedRectangleBorder>(
24              RoundedRectangleBorder(
25                borderRadius: BorderRadius.circular(20.0),
26              ),
27            ),
28          ),
29          onPressed: () {
30            Navigator.push(
31              context,
32              MaterialPageRoute(builder: (context) => page
    ),
33            );
34          },
35          icon: icon,
36          label: Text(
37            text,
38            style: const TextStyle(fontSize: 20),
39          ),
40        ),
41      );
42    }
43  }
```

Codice A.18: Codice del my_choice_button.dart

```
1  import 'package:chatbot/services/auth/auth_service.dart'
       ;
2  import 'package:chatbot/pages/settings_page.dart';
3  import 'package:flutter/material.dart';
4  class MyDrawer extends StatelessWidget {
5    const MyDrawer({
6      super.key,
7    });
8    void logout(){
9      // get auth service
10     final _auth = AuthService();
11     _auth.signOut();
12   }
```

```
13  @override
14  Widget build(BuildContext context){
15    return Drawer(
16      backgroundColor: Theme.of(context).colorScheme.
    background,
17      child: Column(
18        mainAxisAlignment: MainAxisAlignment.
    spaceBetween,
19        children: [
20          Column(
21            children: [
22              //logo
23              DrawerHeader(
24                child: Center(
25                  child: Icon(
26                    Icons.message,
27                    color: Theme.of(context).colorScheme
    .primary,
28                    size: 64,
29                  ),
30                ),
31              ),
32              // home list tile
33              Padding(
34                padding: const EdgeInsets.only(left:
    25.0),
35                child: ListTile(
36                  title: const Text("H O M E"),
37                  leading: const Icon(Icons.home),
38                  onTap: () {
39                    //pop the drawer
40                    Navigator.pop(context);
41                  },
42                ),
43              ),
44              //settings list tile
45              Padding(
46                padding: const EdgeInsets.only(left:
    25.0),
47                child: ListTile(
48                  title: const Text("S E T T I N G S"),
49                  leading: const Icon(Icons.settings),
50                  onTap: () {
```

```
51                    Navigator.pop(context);
52                    Navigator.push(
53                      context,
54                      MaterialPageRoute(
55                        builder: (context) => const
    SettingsPage(),
56                      )
57                    );
58                  },
59                ),
60              ),
61            // //MailChatbot list tile
62            //   Padding(
63            //     padding: const EdgeInsets.only(left:
    25.0),
64            //     child: ListTile(
65            //       title: const Text("M A I L  C H A T
     B O T"),
66            //       leading: const Icon(Icons.mail),
67            //       onTap: () {
68            //         Navigator.pop(context);
69            //         Navigator.push(
70            //           context,
71            //           MaterialPageRoute(
72            //             builder: (context) =>
    MailChatbotPage(),
73            //           )
74            //         );
75            //       },
76            //     ),
77            //   ),
78              // Padding(
79              //   padding: const EdgeInsets.only(left:
    25.0),
80              //   child: ListTile(
81              //     title: const Text("D O C  C H A T B
     O T"),
82              //     leading: const Icon(Icons.
    insert_drive_file),
83              //     onTap: () {
84              //       Navigator.pop(context);
85              //       Navigator.push(
86              //         context,
```

```
87                //          MaterialPageRoute(
88                //            builder: (context) => const
    DocumentChatbotPage(),
89                //          )
90                //        );
91                //      },
92                //    ),
93                // ),
94                // Padding(
95                //   padding: const EdgeInsets.only(left:
    25.0),
96                //   child: ListTile(
97                //     title: const Text("S P E E C H B O
    T"),
98                //     leading: const Icon(Icons.headset),
99                //     onTap: () {
100               //       Navigator.pop(context);
101               //       Navigator.push(
102               //         context,
103               //         MaterialPageRoute(
104               //           builder: (context) =>
    SpeechBotPage(),
105               //         )
106               //       );
107               //     },
108               //   ),
109               // ),
110             ],
111           ),
112           //logout list tile
113           Padding(
114             padding: const EdgeInsets.only(left: 25.0,
    bottom: 25.0),
115             child: ListTile(
116               title: const Text("L O G O U T"),
117               leading: const Icon(Icons.logout),
118               onTap: logout,
119             ),
120           ),
121         ],
122       ),
123     );
124   }
```

```
125 }
```

Codice A.19: Codice del my_drawer.dart

```
1  import 'package:chatbot/services/auth/auth_service.dart'
       ;
2  import 'package:chatbot/services/user/user_string_list.
     dart';
3  import 'package:flutter/material.dart';
4  class MyDropdownMenu extends StatefulWidget{
5    final void Function(String)? selectItem;
6    const MyDropdownMenu({
7      super.key,
8      required this.selectItem,
9    });
10   @override
11   State<MyDropdownMenu> createState() =>
       _MyDropdownMenuState();
12 }
13 class _MyDropdownMenuState extends State<MyDropdownMenu
     >{
14   String selectedItem = '';
15   late void Function(String)? selectItem;
16   List<String> menu = UserStringList().getUserList(
       AuthService().getCurrentUserID());
17   @override
18   void initState(){
19     super.initState();
20     selectItem = widget.selectItem;
21   }
22   @override
23   Widget build(BuildContext context){
24     return Scaffold(
25       body: ListView.builder(
26         itemCount: menu.length ,
27         itemBuilder: (context, index){
28           return GestureDetector(
29             onTap: () {
30               setState(() {
31                 selectedItem = menu[index];
32               });
33               selectItem?.call(menu[index]);
34             },
35             child: Container(
```

```
                color: selectedItem == menu[index]
                  ? Colors.grey.withOpacity(0.3)
                  : Colors.transparent
                ,
                child: ListTile(
                  title: Text(
                    menu[index],
                    style : TextStyle(
                      color: selectedItem == menu[index]
                        ? Colors.blue
                        : Colors.black,
                    ),
                  ),
                  leading: IconButton(
                    icon: const Icon(Icons.close),
                    onPressed: () {
                      UserStringList userList = UserStringList();
                      userList.removeUserElement(AuthService().getCurrentUserID(), index);
                      if(selectedItem == menu[index]){
                        setState(() {
                          selectedItem = '';
                        });
                        selectItem?.call('');
                      }
                    },
                  ),
                ),
              ),
            );
          },
        ),
      );
    }
}
```

Codice A.20: Codice del my_dropdown_menu.dart

```
import 'package:chatbot/components/my_button.dart';
import 'package:chatbot/components/my_dropdown_menu.dart';
import 'package:chatbot/components/my_textfield.dart';
import 'package:chatbot/services/http/my_http.dart';
```

```dart
import 'package:flutter/material.dart';
class MyMailChat extends StatefulWidget{
  const MyMailChat({super.key});
  @override
  State<MyMailChat> createState() => _MyMailChatState();
}
class _MyMailChatState extends State<MyMailChat> {
  final TextEditingController _mailController =
   TextEditingController();
  final TextEditingController _questionController =
   TextEditingController();
  String selectedContext = '';
  void selectContext(String context){
    setState(() {
      selectedContext = context;
    });
  }
  void sendMail(BuildContext context) async{
    MyHtpp myHtpp = MyHtpp();
    String mailText = _mailController.text;
    String questionText = _questionController.text;
    _mailController.clear();
    _questionController.clear();
    try{
      await myHtpp.mailPost(questionText, mailText);
    }
    catch(e){
      showDialog(
        context: context,
        builder: (context) => AlertDialog(
          title: Text(e.toString()),
        )
      );
    }
  }
  @override
  Widget build(BuildContext context){
    return Scaffold(
      appBar: AppBar(
        title: const Text("MailChatbot"),
        backgroundColor: Theme.of(context).colorScheme.
   primary,
      ),
```

```
45      backgroundColor: Theme.of(context).colorScheme.
    secondary,
46      body: Center(
47        child: Row(
48          children: [
49            Expanded(
50              flex: 8,
51              child: Column(
52                mainAxisAlignment: MainAxisAlignment.
    center,
53                children: [
54                  //logo Icon
55                  const Icon(
56                    Icons.email,
57                    size: 60,
58                  ),
59                  const SizedBox(height: 30,),
60                  //text
61                  Text(
62                    "Welcome, insert email and question,
     you will receive the answer by mail",
63                    style: TextStyle(
64                      color: Theme.of(context).
    colorScheme.primary,
65                      fontSize: 16,
66                    ),
67                  ),
68                  const SizedBox(height: 30,),
69                  // email textfield
70                  MyTextField(
71                    hintText: "Email",
72                    obscureText: false,
73                    controller: _mailController,
74                  ),
75                  const SizedBox(height: 10,),
76                  //question textfield
77                  MyTextField(
78                    hintText: "Question",
79                    obscureText: false,
80                    controller: _questionController
81                  ),
82                  const SizedBox(height: 25,),
83                  // send mail button
```

```
                    MyButton(
                      text: "Enter",
                      onTap: () => sendMail(context))
                  ],
                ),
              ),
              Container(
                width: 1.0,
                color: Colors.black,
              ),
              Expanded(
                flex: 2,
                child: MyDropdownMenu(selectItem:
    selectContext,),
              ),
            ],
          ),
        ),
      );
  }
}
```

Codice A.21: Codice del my_mail_chat.dart

```
import 'package:flutter/material.dart';
class MyMessages extends StatelessWidget{
  final List<String> messages;
  const MyMessages({
    super.key,
    required this.messages,
  });
  @override
  Widget build(BuildContext context){
    return ListView.separated(
      padding: const EdgeInsets.all(8.0),
      //reverse: true,
      itemCount: messages.length,
      separatorBuilder: (BuildContext context, int index
    ){
        return const SizedBox(height: 10,);
      },
      itemBuilder: (BuildContext context,int index) {
        Color bubbleColor = index % 2 == 0 ? Colors.blue
    [100]! : Colors.green[100]!;
```

```
19          Alignment alignment = index % 2 == 0 ? Alignment
    .centerLeft : Alignment.centerRight;
20          return Align(
21            alignment: alignment,
22            child: Container(
23              padding: const EdgeInsets.all(10),
24              margin: const EdgeInsets.symmetric(
    horizontal: 20),
25              decoration: BoxDecoration(
26                color: bubbleColor,
27                borderRadius: BorderRadius.circular(20),
28              ),
29              child: Text(
30                messages[index],
31                style: const TextStyle(fontSize: 16),
32              )
33            ),
34          );
35        },
36      );
37    }
38 }
```

Codice A.22: Codice del my_messages.dart

```
1  import 'package:chatbot/services/chat/message.dart';
2  import 'package:flutter/material.dart';
3  import 'package:flutter_tts/flutter_tts.dart';
4  class MySpeechMessages extends StatelessWidget{
5    final List<Message> messages;
6    final FlutterTts _flutterTts = FlutterTts();
7    MySpeechMessages({
8      super.key,
9      required this.messages,
10   });
11   Future<void> speak(String text, BuildContext context)
      async {
12     try{
13       await _flutterTts.setLanguage('it-IT');
14       await _flutterTts.setPitch(1.0);
15       await _flutterTts.setSpeechRate(1.0);
16       await _flutterTts.speak(text);
17     }
18     catch(e){
```

```
19        showDialog(
20          context: context,
21          builder: (context) => AlertDialog(
22            title: Text(e.toString()),
23          )
24        );
25      }
26    }
27    Future<void> pause() async {
28      await _flutterTts.pause();
29    }
30    @override
31    Widget build(BuildContext context){
32      return Scaffold(
33        body: ListView.builder(
34          itemCount: messages.length,
35          itemBuilder: (context,index){
36            Message message = messages[index];
37            return message.isVoice
38              ? VoiceMessageItem(message: message, speak:
     speak, pause: pause)
39              : TextMessageItem(message: message);
40          }
41        ),
42      );
43    }
44  }
45  class VoiceMessageItem extends StatelessWidget {
46    final Message message;
47    final Function(String,BuildContext) speak;
48    final Function pause;
49    const VoiceMessageItem({
50      super.key,
51      required this.message,
52      required this.speak,
53      required this.pause,
54    });
55    @override
56    Widget build(BuildContext context){
57      return ListTile(
58        title: Row(
59          children: [
60            const Icon(Icons.keyboard_voice),
```

```
61            const SizedBox(width: 8,),
62            Expanded(
63              child: Container(
64                decoration: BoxDecoration(
65                  color: Colors.grey[200],
66                  borderRadius: BorderRadius.circular(8),
67                ),
68                padding: EdgeInsets.all(8),
69                child: Row(
70                  children: [
71                    const Expanded(
72                      child: Text("Messaggio vocale"),
73                    ),
74                    const SizedBox(width: 8,),
75                    IconButton(
76                      onPressed: () => speak(message.text,
    context),
77                      icon: Icon(Icons.play_arrow),
78                    ),
79                    IconButton(
80                      onPressed: () => pause(),
81                      icon: const Icon(Icons.pause),
82                    )
83                  ],
84                ),
85              ),
86            ),
87          ],
88        ),
89        //onTap: () => speak(message.text,context),
90      );
91    }
92  }
93  class TextMessageItem extends StatelessWidget{
94    final Message message;
95    const TextMessageItem({
96      super.key,
97      required this.message,
98    });
99    @override
100   Widget build(BuildContext context){
101     return ListTile(
102       title: Text(message.text),
```

```
103     );
104   }
105 }
```

Codice A.23: Codice del my_speech_messages.dart

```
1  import 'dart:convert';
2  import 'package:chatbot/components/my_dropdown_menu.dart
       ';
3  import 'package:chatbot/components/my_speech_messages.
       dart';
4  import 'package:chatbot/services/auth/auth_service.dart'
       ;
5  import 'package:chatbot/services/chat/message.dart';
6  import 'package:chatbot/services/http/my_http.dart';
7  import 'package:flutter/material.dart';
8  import 'package:speech_to_text/speech_recognition_result
       .dart';
9  import 'package:speech_to_text/speech_to_text.dart';
10 class MyTextToSpeech extends StatefulWidget{
11   const MyTextToSpeech({super.key});
12   @override
13   State<MyTextToSpeech> createState() =>
       _MyTextToSpeechState();
14 }
15 class _MyTextToSpeechState extends State<MyTextToSpeech
       >{
16   final SpeechToText _speechToText = SpeechToText();
17   String _speech = '';
18   bool isListening = false;
19   final List<Message> _messages = [];
20   String _context = '';
21   @override
22   void initState(){
23     super.initState();
24     _initSpeech();
25     String text = "Sono il tuo Speechbot, registra un'
       audio per cominciare a chattare";
26     Message message = Message(text: text, isVoice: true)
       ;
27     _messages.add(message);
28   }
29   void _initSpeech() async{
30     await _speechToText.initialize();
```

```
31      setState(() {});
32    }
33    void selectContext(String selectedContext){
34      setState(() {
35        _context = selectedContext;
36      });
37    }
38    void _startListening() async {
39      try{
40        if(!isListening){
41          await _speechToText.listen(onResult:
       _onSpeechResult);
42          setState(() {
43            isListening = true;
44          });
45        }
46      }
47      catch(e){
48        showDialog(
49          context: context,
50          builder: (context) => AlertDialog(
51            title: Text(e.toString()),
52          )
53        );
54      }
55    }
56    void _stopListening() async {
57      try{
58        if(isListening){
59          await _speechToText.stop();
60          setState(() {
61            isListening = false;
62          });
63        }
64      }
65      catch(e){
66        showDialog(
67          context: context,
68          builder: (context) => AlertDialog(
69            title: Text(e.toString()),
70          )
71        );
72      }
```

```
73      Message message = Message(text: _speech, isVoice:
    false);
74      setState(() {
75        _messages.add(message);
76      });
77      await _post();
78    }
79    void _onSpeechResult(SpeechRecognitionResult result){
80      try{
81        setState(() {
82        _speech = result.recognizedWords;
83        });
84      }
85      catch(e){
86        showDialog(
87          context: context,
88          builder: (context) => AlertDialog(
89            title: Text(e.toString()),
90          )
91        );
92      }
93    }
94    Future<void> _post() async {
95      MyHtpp myHtpp = MyHtpp();
96      try{
97        final response = await myHtpp.post(_speech,
    AuthService().getCurrentUserID(), _context);
98        final json = jsonDecode(response.body) as Map<
    String,dynamic>;
99        String text = json['message'];
100       Message message = Message(text: text, isVoice:
    true);
101       setState(() {
102         _speech = "";
103         _messages.add(message);
104       });
105     }
106     catch(e){
107     showDialog(
108       context: context,
109       builder: (context) => AlertDialog(
110         title: Text(e.toString()),
111       )
```

```
112       );
113     }
114   }
115   @override
116   Widget build(BuildContext contex){
117     return Scaffold(
118       appBar: AppBar(
119         title: const Text("SpeechBot"),
120       ),
121       body: Center(
122         child: Row(
123           children: [
124             Expanded(
125               flex: 8,
126               child :Column(
127                 children: [
128                   Expanded(
129                     child: Column(
130                       children: [
131                         Flexible(child: MySpeechMessages
    (messages: _messages)),
132                         const Divider(height: 1.0,),
133                         FloatingActionButton(
134                           onPressed:
135                             !isListening ?
    _startListening : _stopListening,
136                           tooltip: 'Rec',
137                           child: Icon(isListening ?
    Icons.mic_off : Icons.mic),
138                         ),
139                       ],
140                     ),
141                   ),
142                 ],
143               ),
144             ),
145             Container(
146               width: 1.0,
147               color: Colors.black,
148             ),
149             Expanded(
150               flex: 2,
151               child: MyDropdownMenu(selectItem:
```

```
    selectContext,),
152             ),
153           ],
154         ),
155       ),
156     );
157   }
158 }
```

Codice A.24: Codice del my_text_to_speech.dart

```
1 import 'package:flutter/material.dart';
2 class MyTextField extends StatelessWidget {
3   final String hintText;
4   final bool obscureText;
5   final TextEditingController controller;
6   const MyTextField({
7     super.key,
8     required this.hintText,
9     required this.obscureText,
10    required this.controller,
11  });
12  @override
13  Widget build(BuildContext context){
14    return Padding(
15      padding: const EdgeInsets.symmetric(horizontal:
    25.0),
16      child: TextField(
17        obscureText: obscureText,
18        controller: controller,
19        decoration: InputDecoration(
20          enabledBorder: OutlineInputBorder(
21            borderSide: BorderSide(color: Theme.of(
    context).colorScheme.tertiary)
22          ),
23          focusedBorder: OutlineInputBorder(
24            borderSide: BorderSide(color: Theme.of(
    context).colorScheme.primary)
25          ),
26          fillColor: Theme.of(context).colorScheme.
    secondary,
27          filled: true,
28          hintText: hintText,
```

```
29            hintStyle: TextStyle(color: Theme.of(context).
    colorScheme.primary)
30          ),
31        ),
32      );
33    }
34 }
```

Codice A.25: Codice del my_textfield.dart

```
1  import 'package:chatbot/components/my_chat.dart';
2  import 'package:chatbot/services/doc_uploader/
     my_pdf_uploader.dart';
3  import 'package:flutter/material.dart';
4  class PdfPageBody extends StatefulWidget{
5    const PdfPageBody({super.key});
6    @override
7    State<PdfPageBody> createState() => _PdfPageBodyState
       ();
8  }
9  class _PdfPageBodyState extends State<PdfPageBody>{
10   String selectedItem = '';
11   void selectItem(String item){
12     setState(() {
13       selectedItem = item;
14     });
15   }
16   @override
17   Widget build(BuildContext context){
18     return Scaffold(
19       body: Center(
20         child: Row(
21           children: [
22             Expanded(
23               flex: 8,
24               child: const MyChatBot()
25             ),
26             Container(
27               width: 1,
28               color: Colors.black,
29             ),
30             Expanded(
31               flex: 2,
32               child: const MyPdfUploader(),
```

```
              )
            ],
          ),
        ),
      );
    }
}
```

Codice A.26: Codice del pdf_page_body.dart

A.3.1 Servizi

```dart
import 'package:chatbot/pages/choice_page.dart';
import 'package:chatbot/services/auth/login_or_register.
    dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
class AuthGate extends StatelessWidget {
  const AuthGate({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder(
        stream: FirebaseAuth.instance.authStateChanges()
    ,
        builder : (context, snapshot) {
          // user logged in
          if( snapshot.hasData ){
            return const ChoicePage();
          }
          // user is not logged in
          else{
            return const LoginOrRegister();
          }
        }
      ),
    );
  }
}
```

Codice A.27: Codice del auth_gate.dart

```dart
import 'package:firebase_auth/firebase_auth.dart';
```

```
class AuthService {
  // Instance of auth
  final FirebaseAuth _auth = FirebaseAuth.instance;
  //sign in
  Future<UserCredential> signInWithEmailPassword(String email,password) async{
    try{
      UserCredential userCredential = await _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return userCredential;
    } on FirebaseAuthException catch(e){
      throw Exception(e.code);
    }
  }
  // sign up
  Future<UserCredential> signUpWithEmailPassword( String email, password) async {
    try{
      UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password
      );
      return userCredential;
    } on FirebaseAuthException catch(e) {
      throw Exception(e.code);
    }
  }
  // sign out
  Future<void> signOut() async {
    return await _auth.signOut();
  }
  String getCurrentUserID(){
    User? user = _auth.currentUser;
    if(user != null){
      return user.uid;
    }
    else{
      throw Exception("User not logged in");
    }
```

```
41   }
42 }
```

Codice A.28: Codice del auth_service.dart

```
1  import 'package:chatbot/pages/login_page.dart';
2  import 'package:chatbot/pages/register_page.dart';
3  import 'package:flutter/material.dart';
4  class LoginOrRegister extends StatefulWidget {
5    const LoginOrRegister({super.key});
6    @override
7    State<LoginOrRegister> createState() =>
       _LoginOrRegisterState();
8  }
9  class _LoginOrRegisterState extends State<
       LoginOrRegister> {
10   // initially show login page
11   bool showLoginPage = true;
12   // toggle beetween login and register page
13   void togglePages() {
14     setState(() {
15       showLoginPage = !showLoginPage;
16     });
17   }
18   @override
19   Widget build(BuildContext context) {
20     if(showLoginPage) {
21       return LoginPage(onTap: togglePages,);
22     }
23     else{
24       return RegisterPage(onTap: togglePages,);
25     }
26   }
27 }
```

Codice A.29: Codice del login_or_register.dart

```
1  class Message {
2      final String text;
3      final bool isVoice;
4      Message({required this.text, required this.isVoice})
       ;
5    }
```

Codice A.30: Codice del message.dart

```dart
import 'package:chatbot/components/my_textfield.dart';
import 'package:chatbot/services/auth/auth_service.dart';
import 'package:chatbot/services/http/my_http.dart';
import 'package:flutter/material.dart';
class MyHtmlUploader extends StatelessWidget{
  final TextEditingController _urlsController = TextEditingController();
  final TextEditingController _contextNameController = TextEditingController();
  MyHtmlUploader({super.key});
  void sendUrls(BuildContext context) async{
    MyHtpp myHtpp = MyHtpp();
    String urls = _urlsController.text;
    String contextName = _contextNameController.text;
    String userId = AuthService().getCurrentUserID();
    if(urls.isEmpty || contextName.isEmpty){
      showDialog(
        context: context,
        builder: (context) => AlertDialog(
          title: const Text("One of the fields is empty"),
        )
      );
      return;
    }
    try{
      await myHtpp.sendUrl(urls,userId,contextName);
    }
    catch(e){
      showDialog(
        context: context,
        builder: (context) => AlertDialog(
          title: Text(e.toString()),
        )
      );
    }
  }
  @override
  Widget build(BuildContext context){
    return Scaffold(
      body: Column(
```

```
39          crossAxisAlignment: CrossAxisAlignment.start,
40          children: [
41            Container(
42              height: 1.0,
43              color: Colors.black,
44            ),
45            Padding(
46              padding: const EdgeInsets.all(16.0),
47              child : const Text("Insert urls following by
     a space"),
48            ),
49            const SizedBox(height: 10,),
50            Expanded(
51              child: Padding(
52                padding: const EdgeInsets.all(16.0),
53                child: TextField(
54                  controller: _urlsController,
55                  maxLines: null,
56                  decoration: const InputDecoration(
57                  hintText: 'Enter URLs(one per line)',
58                  ),
59                ),
60              ),
61            ),
62            const SizedBox(height: 20,),
63            Container(
64              height: 2.0,
65              color: Colors.black,
66            ),
67            const SizedBox(height: 40.0,),
68            Padding(
69              padding: const EdgeInsets.symmetric(
    horizontal: 16.0),
70              child: MyTextField(
71                hintText: "Type context name",
72                obscureText: false,
73                controller: _contextNameController,
74              ),
75            ),
76            Container(
77              height: 1.0,
78              color: Colors.black,
79            ),
```

```
80            Padding(
81              padding: const EdgeInsets.all(16.0),
82              child: ElevatedButton(
83                onPressed: () => sendUrls(context),
84                child: const Text('Send'),
85              ),
86            ),
87          ],
88        ),
89      );
90    }
91  }
```

Codice A.31: Codice del my_html_uploader.dart

```
1  import 'dart:convert';
2  import 'package:chatbot/components/my_textfield.dart';
3  import 'package:chatbot/services/auth/auth_service.dart'
       ;
4  import 'package:chatbot/services/http/my_http.dart';
5  import 'package:chatbot/services/user/user_string_list.
       dart';
6  import 'package:file_picker/file_picker.dart';
7  import 'package:flutter/material.dart';
8  class MyPdfUploader extends StatefulWidget {
9    const MyPdfUploader({
10     super.key,
11   });
12   @override
13   State<MyPdfUploader> createState() =>
       _MyPdfUploaderState();
14 }
15 class _MyPdfUploaderState extends State<MyPdfUploader>{
16   final List<PlatformFile> _uploadedFiles = [];
17   final TextEditingController _nameListController =
       TextEditingController();
18   Future<void> _pickPDF(BuildContext context) async {
19     try{
20       FilePickerResult? result = await FilePicker.
       platform.pickFiles(
21         allowMultiple: true,
22         type: FileType.custom,
23         allowedExtensions: ['pdf'],
24       );
```

```
25        if (result != null) {
26          setState(() {
27            _uploadedFiles.addAll(result.files);
28          });
29        }
30        }
31      catch(e){
32        showDialog(
33          context: context,
34          builder: (context) => AlertDialog(
35            title: Text(e.toString()),
36          )
37        );
38      }
39    }
40    void _processPdfs(BuildContext context) async{
41      if(_uploadedFiles.isEmpty || _nameListController.
      text.isEmpty)
42        showDialog(
43          context: context,
44          builder: (context) => AlertDialog(
45            title: Text("Files or name field missing"),
46          )
47        );
48        else{
49          MyHtpp myHtpp = MyHtpp();
50          try{
51            String directoryName = _nameListController.
      text;
52            _nameListController.clear();
53            final response = await myHtpp.sendFiles(
      _uploadedFiles,directoryName);
54            final responseData = await response.stream.
      bytesToString();
55            Map<String,dynamic> decodedResponse = json.
      decode(responseData);
56            showDialog(
57              context: context,
58              builder: (context) => AlertDialog(
59              title: Text(decodedResponse['message']),
60              )
61            );
62            UserStringList userList = UserStringList();
```

```
63            AuthService authService = AuthService();
64            final userId = authService.getCurrentUserID();
65            userList.addUserElement(userId, directoryName)
    ;
66          }
67          catch(e){
68            showDialog(
69            context: context,
70            builder: (context) => AlertDialog(
71            title: Text(e.toString()),
72          )
73            );
74          }
75        }
76    }
77    void _removeFile(int index){
78      setState(() {
79        _uploadedFiles.removeAt(index);
80      });
81    }
82    @override
83    Widget build(BuildContext context){
84      return Scaffold(
85        body: Center(
86          child: Column(
87            mainAxisAlignment: MainAxisAlignment.center,
88            children: [
89              Container(
90                height: 1,
91                color: Colors.black,
92              ),
93              Padding(
94                padding: const EdgeInsets.all(30.0),
95                child: ElevatedButton(
96                  onPressed: () => _pickPDF(context),
97                  style: ButtonStyle(
98                    fixedSize: MaterialStateProperty.all<
    Size>(
99                      const Size(200, 50),
100                   ),
101                 ),
102                 child: const Text(
103                   "Load PDF files",
```

```
                  style: TextStyle(color: Colors.black),
                ),
              ),
            ),
            const SizedBox(height: 20,),
            Expanded(
              child: ListView.builder(
                itemCount: _uploadedFiles.length,
                itemBuilder: (context, index){
                  return Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: ListTile(
                      leading: const Icon(Icons.
picture_as_pdf),
                      title: Text(_uploadedFiles[index].
name),
                      trailing: IconButton(
                        icon: const Icon(Icons.close),
                        onPressed: () => _removeFile(
index),
                      ),
                    ),
                  );
                },
              ),
            ),
            Container(
              height: 1,
              color: Colors.black,
            ),
            MyTextField(
              hintText: "Type your context name",
              obscureText: false,
              controller: _nameListController,
            ),
            Padding(
              padding: const EdgeInsets.all(30.0),
              child: ElevatedButton(
                onPressed: () => _processPdfs(context),
                style: ButtonStyle(
                  fixedSize: MaterialStateProperty.all<
Size>(
                    const Size(200, 50),
```

```
                    ),
                  ),
                  child: const Text(
                    "Process Pdfs",
                    style: TextStyle(color: Colors.black),
                  ),
                ),
              ),
            ],
          ),
        ),
      );
  }
}
```

Codice A.32: Codice del my_pdf_uploader.dart

```
import 'dart:convert';
import 'package:chatbot/services/auth/auth_service.dart';
import 'package:file_picker/file_picker.dart';
import 'package:http/http.dart' as http;
class MyHtpp {
  String url = "localhost:8000";
  Future<http.Response> post(String message,String userId, String context) async {
    final response = await http.post(
      Uri.http(url,'/'),
      headers: <String,String> {
        'Content-type' : 'application/json; charset=UTF-8',
      },
      body: jsonEncode(<String,String>{
        'function' : "message",
        'message': message,
        'userId' : userId,
        'context' : context,
      }),
    );
    if(response.statusCode == 200){
      return response;
    }
    else{
      throw Exception('Failed to get response');
```

```
25      }
26    }
27    Future<http.Response> sendUrl(String message,String
       userId, String context) async {
28      final response = await http.post(
29        Uri.http(url,'/url'),
30        headers: <String,String> {
31          'Content-type' : 'application/json; charset=UTF
       -8',
32        },
33        body: jsonEncode(<String,String>{
34          'message': message,
35          'userId' : userId,
36          'context' : context,
37        }),
38      );
39      if(response.statusCode == 200){
40        return response;
41      }
42      else{
43        throw Exception('Failed to get response');
44      }
45    }
46    Future<http.Response> getUserDirectory(String userId)
       async {
47      final response = await http.post(
48        Uri.http(url,'/directory'),
49        headers: <String,String> {
50          'Content-type' : 'application/json; charset=UTF
       -8',
51        },
52        body: jsonEncode(<String,String>{
53          'userId' : userId,
54        }),
55      );
56      if(response.statusCode == 200){
57        return response;
58      }
59      else{
60        throw Exception('Failed to get response');
61      }
62    }
63    Future<http.Response> createUserDirectory(String
```

```
 userId) async {
64    final response = await http.post(
65      Uri.http(url,'/create_directory'),
66      headers: <String,String> {
67        'Content-type' : 'application/json; charset=UTF
   -8',
68      },
69      body: jsonEncode(<String,String>{
70        'userId' : userId,
71      }),
72    );
73    if(response.statusCode == 200){
74      return response;
75    }
76    else{
77      throw Exception('Failed to get response');
78    }
79  }
80  Future<http.Response> mailPost(String message,String
    mailReceiver) async{
81    final response = await http.post(
82      Uri.http(url,'/'),
83      headers: <String,String> {
84        'Content-type' : 'application/json; charset=UTF
   -8',
85      },
86      body: jsonEncode(<String,String>{
87        'function' : "mail",
88        'message': message,
89        'mail' : mailReceiver,
90      }),
91    );
92    if(response.statusCode == 200){
93      return response;
94    }
95    else{
96      throw Exception('Failed to get response');
97    }
98  }
99  Future<http.StreamedResponse> sendFiles(List<
    PlatformFile> uploadedFiles, String directoryName)
    async{
100   var formData = http.MultipartRequest('POST',Uri.http
```

```
    (url,'/upload'));
101   for(int i = 0; i < uploadedFiles.length;i++){
102     PlatformFile file = uploadedFiles[i];
103     formData.files.add(http.MultipartFile.fromBytes(
104       "files",
105       file.bytes!,
106       filename: file.name,
107     ));
108   }
109   AuthService authService = AuthService();
110   String userId = authService.getCurrentUserID();
111   Map<String,dynamic> jsonData = {
112     'directory_name' : directoryName,
113     'userID' : userId,
114   };
115   formData.files.add(http.MultipartFile.fromString(
116     "jsonFile",
117     jsonEncode(jsonData),
118     filename: "data.json"
119   ));
120   final response = await http.Client().send(formData);
121   if(response.statusCode == 200){
122     return response;
123   }
124   else{
125     throw Exception("Faield to get response");
126   }
127  }
128 }
```

Codice A.33: Codice del my_http.dart

```
1 import 'dardt:convert';
2 import 'package:chatbot/services/http/my_dhdtdtp.dart';
3 class UserSdtdringList {
4   static final Map<String, List<String>> _userdLdidsts =
      {};
5   static Future<void> initializeUserList(String userdId)
      adsync {
6     MyHtpp myHtpp d=d MyHtpp()d;d
7     try{
8       final response = await myHtpp.getUserDirectodrdy(
    userId);
```

```
9         final json = jsonDecode(response.body) as Map<
    Strindgd,dynamic>;
10        List<String> directoryList = List<String>.from(
    json['directodrdy_list']);
11        _userLists[userId] = dirdedctoryList;
12     d}catch(e) {
13        throw Exception(e.tdodStringd());
14     }
15   }
16   static Future<void> createUserList(String userdId)
     adsync {
17     MyHtpp myHtpp d=d MyHtpp()d;d
18     try{
19        await myHtpp.createUserDirectodrdy(userId);
20        //final json = jsonDecode(response.body) as Map<
    Strindgd,dynamic>;
21        _userLists[usdedrId] = [];
22     d}catch(e) {
23        throw Exception(e.tdodStringd());
24     }
25   }
26   List<String> getUserList(Stridndg userId){
27     List<String>d dlist = [];
28     if(_userLists.containsKedy(userIdd)){
29        list = _userListds[userdIdd]!;
30     }
31     rdeturnd dlistd;d
32   }
33   void addUserElement(String userId, String
     diredcdtoryName){
34     _userLists[userId]!.add(diredcdtorydNdame);
35   }
36   void removeUserElement(String userId, dint index){
37     _userLists[userId]!.removdeAt(didnddex);
38   }
39 }
```

Codice A.34: Codice del user_string_list.dart