UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione



TESI DI LAUREA

Progettazione e realizzazione di case study per il confronto di Large Language Model differenti nella gestione di testi, immagini e software

Design and implementation of case studies to compare different Large Language Models in the text, image and software management

Relatore

Prof. Domenico Ursino

Candidato

Leon Ursu

ANNO ACCADEMICO 2025-2026

Non c'era scelta se non essere pionieri

Margaret Hamilton

Sommario

L'Intelligenza Artificiale Generativa, negli ultimi anni, sta assumendo un'importanza crescente nell'ambito della ricerca e dello sviluppo tecnologico mondiale. Questa tecnologia sta acquisendo un rilevante peso politico e sociale, attirando sempre più investimenti e stimolando movimenti legislativi per la sua regolamentazione.

La presente tesi è stata realizzata con l'intento di mettere in luce alcuni dei possibili casi d'uso in cui applicare la tecnologia dell'Intelligenza Artificiale Generativa, sia nell'ambito lavorativo che in quello didattico. L'analisi si è concentrata sullo sviluppo di tre distinti case study riguardanti la generazione di testo, di immagini e di codice; ciò ci ha permesso di apprendere che questa tecnologia ha un alto potenziale d'uso ma necessita ancora della supervisione umana, in quanto i risultati ottenuti risultano privi del tassello finale indispensabile per un funzionamento corretto.

Keyword: Intelligenza Artificiale, ChatGPT, Gemini, Claude, Sonnet, Codex, Intelligenza Artificiale Generativa, Super Intelligence, SI

Indice

Introduzione **1**

1 Introduzione alla Generative AI **3**
1.1 Definizione dell'Intelligenza Artificiale Generativa . . . 3
1.2 Evoluzione storica dell'Intelligenza Artificiale classica fino ai Large Language Model . . . 3
1.3 Panoramica sui meccanismi di funzionamento . . . 5
1.3.1 Parametri di un modello . . . 5
1.3.2 Parametri totali vs parametri attivi . . . 5
1.3.3 Classificazione dei modelli AI . . . 6
1.3.4 Limiti al numero di parametri utilizzati . . . 6
1.4 I modelli più usati e alternative open-source . . . 6
1.5 Campi di applicazione e impatto sociale . . . 7

2 Descrizione degli LLM considerati **9**
2.1 OpenAI - ChatGPT e l'ecosistema Codex . . . 9
2.1.1 Origini e contesto tecnologico . . . 9
2.1.2 Architettura e caratteristiche tecniche . . . 11
2.1.3 Interfaccia utente . . . 11
2.1.4 Modalità di input e funzionalità integrate . . . 13
2.2 Google - Gemini . . . 14
2.2.1 Origini e contesto tecnologico . . . 14
2.2.2 Architettura e caratteristiche tecniche . . . 15
2.2.3 Interfaccia utente . . . 15
2.2.4 Modalità di input e funzionalità integrate . . . 16
2.3 Anthropic - Claude . . . 16
2.3.1 Origini e contesto tecnologico . . . 16
2.3.2 Architettura e caratteristiche tecniche . . . 17
2.3.3 Interfaccia utente . . . 17
2.3.4 Modalità di input e funzionalità integrate . . . 18
2.3.5 Analisi comparativa dei modelli considerati . . . 18

3 Confronto degli LLM nella gestione dei testi **20**
3.1 Introduzione e definizione del problema . . . 20
3.1.1 Descrizione del problema testuale . . . 20

3.1.2 Obiettivo della sperimentazione . . . 20
3.1.3 Criteri di valutazione . . . 21
3.2 Metodologia di analisi . . . 21
3.2.1 Selezione dei modelli di Intelligenza Artificiale . . . 21
3.2.2 Progettazione del prompt iniziale . . . 22
3.2.3 Vincoli progettuali dei prompt secondari . . . 22
3.3 Esecuzione del test . . . 22
3.3.1 Scenario di test con Gemini . . . 22
3.3.2 Scenario di test con ChatGPT . . . 29
3.4 Analisi comparativa dei risultati . . . 36
3.5 Discussione, limiti e linee guida pratiche . . . 36
3.5.1 Limiti dell'Intelligenza Artificiale nella diagnostica meccanica . . . 37
3.5.2 Linee guida pratiche . . . 37
3.6 Conclusione . . . 38

4 Confronto degli LLM nella gestione delle immagini 39
4.1 Introduzione e definizione del problema . . . 39
4.1.1 Artefatti visivi delle immagini . . . 39
4.1.2 Obiettivo della sperimentazione . . . 41
4.1.3 Progettazione del prompt iniziale . . . 42
4.2 Esecuzione del test . . . 43
4.2.1 Scenario di test con Gemini . . . 43
4.2.2 Scenario di test con ChatGPT . . . 45
4.3 Analisi comparativa . . . 50
4.3.1 Confronto delle prestazioni e accuratezza . . . 50
4.3.2 Limiti riscontrati ed errori ricorrenti . . . 51
4.4 Conclusioni generali . . . 51

5 Confronto degli LLM nella gestione del codice 53
5.1 Obiettivo dello studio e metodologia di approccio . . . 53
5.1.1 Obiettivo del confronto . . . 53
5.1.2 Metodologia di analisi . . . 53
5.1.3 Criterio di selezione dei siti target . . . 54
5.2 Scenario di test per il sito di e-commerce statico per training . . . 57
5.2.1 Analisi degli obiettivi e complessità del target . . . 57
5.2.2 Sviluppo dello scraper con il modello Codex OpenAI . . . 57
5.2.3 Sviluppo dello scraper con il modello Claude Sonnet . . . 60
5.2.4 Valutazione comparativa sul sito target . . . 65
5.3 Scenario di test per il sito di e-commerce di giardinaggio . . . 65
5.3.1 Analisi degli obiettivi e complessità del target . . . 65
5.3.2 Sviluppo dello scraper con il modello Codex OpenAI . . . 66
5.3.3 Sviluppo dello scraper con il modello Claude Sonnet . . . 71
5.3.4 Valutazione comparativa sul sito target . . . 79
5.4 Scenario di test per il sito di e-commerce di abbigliamento . . . 79
5.4.1 Analisi degli obiettivi e complessità del target . . . 79
5.4.2 Sviluppo dello scraper con il modello Codex OpenAI . . . 80
5.4.3 Sviluppo dello scraper con il modello Claude Sonnet . . . 89
5.4.4 Valutazione comparativa sul sito target . . . 101
5.5 Discussione e linee guida pratiche . . . 101

6 Discussione **103**
6.1 Introduzione alla discussione 103
6.2 Discussione sul case study per il testo 103
6.3 Discussione sul case study per le immagini 103
6.4 Discussione sul case study per il codice 104

Conclusioni **105**

Bibliografia **107**

Sitografia **109**

Ringraziamenti **110**

Elenco delle figure

1.1 Grafico delle ricerche sui sistemi di Intelligenza Artificiale 7

2.1 Interfaccia di ChatGPT 12
2.2 Interfaccia di Codex 13
2.3 Interfaccia di Gemini 15
2.4 Interfaccia di Claude 18

4.1 Immagine contenente un esempio di pseudo testo 41

Elenco delle tabelle

2.1 Capacità e punti di forza dei sistemi di Intelligenza Artificiale considerati . . 19

3.1 Valutazione qualitativa dei modelli di Intelligenza Artificiale 36
3.2 Valutazione quantitativa dei modelli di Intelligenza Artificiale 36

4.1 Valutazione quantitativa dei modelli di Intelligenza Artificiale 50

5.1 Caratteristiche tecnologiche dei siti considerati 54
5.2 Confronto dei requisti tra i modelli utilizzati 65
5.3 Confronto dei requisti tra i modelli utilizzati 79
5.4 Confronto dei requisti tra i modelli utilizzati 102

Introduzione

Negli ultimi anni, l'Intelligenza Artificiale Generativa, grazie a investimenti ingenti, sta attuando profonde trasformazioni sia nelle modalità di apprendimento sia nei contesti lavorativi. Come emerge dall'analisi dei dati pubblicamente disponibili, il settore dell'Information Technology (IT), risulta quello più coinvolto in tale trasformazione visto il suo potenziale. Gli LLM sono, infatti, in grado di generare codice sorgente in pochi secondi a partire da un input espresso in linguaggio naturale; possiamo paragonare questo aspetto all'utilizzo di un linguaggio di livello di astrazione maggiore rispetto ai classici linguaggi di programmazione ad alto livello.

Lo sviluppo di un software risulta radicalmente trasformato. Negli anni antecedenti all'utilizzo dei generatori di codice, osservando un generico diagramma di Gantt del ciclo di produzione di un software, si notava come le fasi di scrittura del codice e del testing consumavano la maggior parte delle risorse temporali di un progetto. Al giorno d'oggi, osservando lo stesso diagramma, si potrebbero notare delle singolarità interessanti: la sezione di sviluppo è diminuita, mentre la sezione di testing e di revisione del codice generato è aumentata.

Le piattaforme come Claude, ChatGPT e Gemini sono diventate dei punti di riferimento importanti nello sviluppo del software in quanto forniscono gli strumenti di generazione attualmente più diffusi. La loro ampia diffusione è causata sia da investimenti di marketing effettuati dalle rispettive aziende, sia dal continuo rilascio di versioni aggiornate dei modelli, sui quali i professionisti fanno affidamento.

Vista la rapida diffusione di questi strumenti, ci è sembrato interessante effettuare un'analisi critica e metodologica per valutare il loro potenziale in prima persona. La nostra sperimentazione riguarderà lo sviluppo di tre case study.

Il primo è dedicato alla valutazione del testo e riguarda la risoluzione di un problema di diagnostica meccanica in ambito automobilistico. Attraverso la progettazione di un prompt iniziale, il lavoro si propone di giungere alla soluzione finale mediante l'aggiunta graduale delle nuove informazioni, come, ad esempio, la misurazione delle resistenze elettriche dei componenti mediante un multimetro. Il secondo case study incentrato sulle immagini si svilupperà utilizzando gli strumenti di generazione per creare mockup di interfacce utente da utilizzare in ambienti critici, come, ad esempio, le ambulanze. Il terzo case study riguarderà la generazione di codice per attuare scraping su siti di e-commerce, con l'intento di recuperare nomi e prezzi di prodotti che sono disponibili.

Una volta definiti i requisiti funzionali e non funzionali che si intendono far implementare dai sistemi di Intelligenza Artificiale, gli stessi sistemi verranno messi a confronto verificando

quali requisiti sono stati soddisfatti e quali non lo sono stati. Questo ci permetterà di prendere visione dell'efficacia e del comportamento complessivo dei vari sistemi.

Questa tesi è strutturata come di seguito specificato:

- Nel Capitolo 1 sarà introdotta la definizione dell'Intelligenza Artificiale Generativa; successivamente verrà spiegata la storia, e infine, verranno illustrati i meccanismi di funzionamento di questa tecnologia.
- Nel Capitolo 2 saranno introdotti i modelli di AI Generativa che sono stati da noi considerati.
- Nel Capitolo 3 sarà eseguito il nostro case study relativo al testo. L'analisi riguarderà la diagnostica meccanica ed elettrica di un automobile che presenta un malfunzionamento.
- Nel Capitolo 4 sarà eseguito il nostro case study relativo alla generazione delle immagini, che riguarderà la definizione di interfacce utente per le console centrali all'interno di un'ambulanza.
- Nel Capitolo 5 sarà eseguito il nostro case study relativo alla generazione di codice per lo scraping su tre siti web differenti.
- Nel Capitolo 6 verrà discusso il risultato ottenuto nelle tre sperimentazioni.

CAPITOLO 1

Introduzione alla Generative AI

In questo capitolo procederemo con la spiegazione di cosa sia l'Intelligenza Artificiale Generativa, partendo dalla sua definizione per poi proseguire con la sua evoluzione storica.

Al giorno d'oggi, sembra che tale tecnologia sia molto recente, e in un certo senso lo è; ma, come vedremo, l'idea di Intelligenza Artificiale nasce molto tempo fa.

Definiremo cosa sono i parametri che vengono utilizzati e faremo una panoramica sui diversi tipi di Intelligenza Artificiale utilizzando come termine discriminante il numero di parametri che essa contiene.

Effettuare una valutazione sull'utilizzo dei sistemi di Intelligenza Artificiale è una sfida. Al giorno d'oggi accedere a tale tecnologia è molto semplice: è sufficiente avere una connessione alla rete e un dispositivo di navigazione; per questo motivo i dati dell'utilizzo sono stati ricavati in maniera empirica e cambiano quasi quotidianamente, visto anche la veloce evoluzione di tali sistemi.

Presenteremo una lista di sistemi di Intelligenza Artificiale che sono al momento nel trend e vedremo anche alternative open-source.

Infine, valuteremo come tale tecnologia può essere applicata alla vita quotidiana indipendentemente dal contesto e valuteremo l'impatto sociale che genera.

1.1 Definizione dell'Intelligenza Artificiale Generativa

L'Intelligenza Artificiale Generativa è un sottosistema dei sistemi di Intelligenza Artificiale, in grado di generare principalmente video, immagini, testo e audio.

Il primo utilizzo del termine "Intelligenza Artificiale Generativa" è stato individuato negli anni '90, come cita l'Oxford English Dictionary.

Negli ultimi tempi abbiamo visto una vera e propria rivoluzione tecnologica che porta sia margini di innovazione che sfide in molteplici settori di appartenenza.

L'obiettivo dichiarato dai produttori dei sistemi di Intelligenza Artificiale in cui siamo quotidianamente immersi varia caso per caso. Tuttavia, eseguendo un'analisi delle mission delle principali imprese che operano in questo settore, possiamo individuare un fattore comune, ovvero il bene dell'umanità, tralasciando ovviamente lo scopo di lucro che non viene mai citato.

1.2 Evoluzione storica dell'Intelligenza Artificiale classica fino ai Large Language Model

Il termine "Intelligenza Artificiale" è noto agli esseri umani fin dai miti greci in cui si parlava di creature artificiali nate con un proprio pensiero e volere. Man mano che si prosegue

con la storia, nel XX secolo D.C. il tema dei robot è diventato il capostipite di storie e articoli letterari creati per intrattenimento.

Possiamo affermare che la vera e propria nascita dell'Intelligenza Artificiale, intesa come una macchina pensante, si ha intorno al 1940, dopo che le neuroscienze hanno effettuato delle scoperte in cui si affermava che un cervello umano funziona con una rete neurale governata da impulsi elettrici.

Nacquero numerosi studi che esplorarono molteplici direzioni per cercare informazioni su come poter sviluppare l'Intelligenza Artificiale. Lo stesso Alan Turing nel 1950 ha pubblicato un saggio denominato "Computing machinery and intelligence" in cui speculava sulla possibilità della creazione di tale tecnologia. Questo portò alla nascita del Test di Turing, dove si domanda se una conversazione coerente tra un individuo umano e due entità, un umano e una macchina, attraverso un protocollo di comunicazione, possa determinare se una macchina è in grado di pensare.

Un successo iniziale avvenne tra il 1956 e il 1974 in cui, grazie alle ricerche sull'Intelligenza Artificiale, dei ricercatori riuscirono a far comunicare dei computer in linguaggio naturale e risolvere problemi di algebra, anch'essi descritti in linguaggio naturale. Tuttavia, tali risultati rimasero insufficienti ad attirare fondi per la ricerca e lo sviluppo costante.

L'Intelligenza Artificiale dovette affrontare due "AI Winter" prima di giungere al XXI secolo e attirare nuovi investitori. Tra il 2005 e il 2017 ci fu un massiccio investimento di risorse da parte delle aziende operanti nel settore, spinte dalla possibilità di crescita economica e dalla competitività nei settori di Big Data e Deep Learning. Nonostante tutte le ricerche e risorse spese nel settore dell'Intelligenza Artificiale non si poté tracciare nettamente la frontiera tra l'era prima e dopo la nascita dell'AI.

Possiamo individuare tale frontiera con buona precisione nel 2017, e può essere chiamata "The AI Boom"; in quest'anno nacquero numerosi sistemi di Intelligenza Artificiale; tra i produttori più importanti citiamo: Google, OpenAI e Anthropic.

Nel 2017 Google rilascia un documento chiamato "Attention is all you need"; esso è un'introduzione all'architettura Transformer che rivoluziona le modalità delle valutazioni in base al peso delle parole all'interno di una frase.

Successivamente OpenAI rilascia GPT3 nel 2020, e nel 2022 DeepMind rilascia Gato. I modelli appena citati sono stati addestrati su una grande quantità di dati e possono coprire un grande quantitativo di argomenti.

La nascita del grande interesse per l'Intelligenza Artificiale avviene a novembre del 2022, periodo in cui viene rilasciato ChatGPT da OpenAI. Questa applicazione scala le classifiche e acquisisce 100 milioni di utenti nei primi due mesi dal lancio; per la prima volta un sistema di Intelligenza Artificiale supera la fase di prototipazione e diventa un prodotto consumer.

Il 2023 è un anno in cui le Big Tech entrano in una fase di alta competitività. Microsoft investe miliardi di dollari in OpenAI e integra Copilot nativamente in Windows e Office. Google rilascia Bard a metà del 2023, che poi verrà integrato direttamente nel nuovo Gemini a febbraio 2024. Nel 2023 Anthropic rilascia Claude, ma per basse capacità e poco preavviso da parte dei competitor nel rilascio di nuovi modelli, passa per un momento in secondo piano.

La grande popolarità di ChatGPT ha spinto le Big Tech presenti per il mondo verso una vera e propria corsa al rilascio del proprio modello di Intelligenza Artificiale al mondo consumer nel minor tempo possibile.

Nel corso del 2024 si sono susseguiti diversi rilasci di modelli da parte di molte compagnie; una particolare attenzione ha ricevuto Claude 3, che conquista una posizione importante grazie alla sua capacità di generare risposte di testo meno robotiche e più umane. Proseguendo nello stesso anno, l'accento cade sempre più sui sistemi di Intelligenza Artificiale multimodali e non più sui semplici chatbot che diventano capaci di gestire e creare sempre più tipologie di file.

Un evento importante riguarda l'invenzione degli Agenti basati sui sistemi di Intelligenza Artificiale che riescono a svolgere svariati compiti, ad utilizzare un computer in autonomia e ad effettuare dei task definiti dagli utenti tramite linguaggio naturale, che fino a quel momento, erano impossibili. Questi nuovi sistemi basati sugli agenti vengono ottimizzati sempre con più frequenza.

Lo stato attuale dell'Intelligenza Artificiale è in continuo avanzamento; tale tecnologia si sta progressivamente consolidando nei settori come sostituzione al lavoro ripetitivo; molte aziende si trovano in una fase di riallocazione delle risorse, vista la crescente integrazione dell'Intelligenza Artificiale nei sistemi informatici. Le principali difficoltà riscontrate nell'utilizzo dell'Intelligenza Artificiale nell'automazione completa di un progetto è l'assenza di visione completa di un problema da risolvere; tale fenomeno è presente in molti settori e continua a essere la sfida principale da affrontare; è qui che possiamo individuare una posizione in cui la figura umana non è ancora sostituibile.

Il futuro dei sistemi di Intelligenza Artificiale è imprevedibile; possiamo affermare con buona certezza che tali sistemi continueranno a crescere sia nelle capacità computazionali che nel consumo di materie prime e di energia elettrica.

1.3 Panoramica sui meccanismi di funzionamento

Così come la mente umana è un insieme di neuroni che comunicano tramite impulsi elettrici viaggianti sulle sinapsi, un modello di Intelligenza Artificiale moderna è costruito con un concetto affine.

La rete neurale è un grafo formato da nodi che sono connessi tramite archi ed è composta da tre strati principali. Lo strato dell'input è il punto d'ingresso dell'informazione che deve essere elaborata attraverso la rete neurale. Lo strato dell'output è il punto di uscita dell'informazione elaborata attraverso la rete. Tra i due strati fondamentali vi sono una serie di strati nascosti, chiamati "hidden layer". Essi sono direttamente responsabili dell'elaborazione delle informazioni che sono state fornite alla rete neurale. Il risultato prodotto da ogni strato viene trasmesso agli strati successivi fino ad arrivare all'uscita della rete.

1.3.1 Parametri di un modello

Uno dei meccanismi principali che permettono il funzionamento dell'Intelligenza Artificiale sono il numero di parametri che ogni modello possiede. Tale numero rappresenta il quantitativo di archi e nodi da cui la rete neurale è formata. Possiamo dividere i parametri di un sistema in due categorie principali:

- *Pesi*: determinano l'importanza del collegamento tra due nodi consecutivi. Hanno solitamente un valore numerico. Se la correlazione tra due nodi è molto alta, come ad esempio tra le parole "casa" e "tetto", il valore numerico sarà maggiore.
- *Bias*: determina un valore fisso per ogni nodo. Il suo valore si somma al valore numerico che arriva al nodo. Funge da amplificatore del risultato. Tale valore viene appreso durante l'addestramento di un sistema di Intelligenza Artificiale e non viene modificato durante la sua esecuzione, ma soltanto durante un successivo addestramento.

1.3.2 Parametri totali vs parametri attivi

Un'altra distinzione importante per i sistemi di Intelligenza Artificiale è quella tra parametri totali e parametri attivi. In particolare:

- *Parametri Totali*: determinano il numero complessivo di nodi e archi di cui un modello di Intelligenza Artificiale è dotato. Il numero è determinato dall'addestramento della rete neurale: maggiori sono le informazioni e i pattern che ha appreso e maggiore sarà il numero di parametri.
- *Parametri Attivi*: un modello di Intelligenza Artificiale, per rispondere ad un prompt, non attiva l'intero set di parametri che possiede, visto che questi possono superare le centinaia di milioni. Esso infatti attiva solamente una piccola porzione di tale set, che è specializzata nell'elaborazione degli argomenti e della richiesta estratti dal linguaggio naturale del prompt.

1.3.3 Classificazione dei modelli AI

I modelli di Intelligenza Artificiale possono essere classificati in:

- *Unimodali*: Sono sistemi che possono gestire un solo tipo di dato: testo, immagini o video. Vengono adoperati soprattutto nei vecchi chatbot di elaborazione del testo.
- *Multimodali*: Sono sistemi che gestiscono generalmente un'ampia tipologia di dati e possono accettare più tipologie di file nello stesso prompt.

1.3.4 Limiti al numero di parametri utilizzati

Sebbene il numero di parametri utilizzati dai modelli più famosi di Intelligenza Artificiale non sia disponibile direttamente, visto che questo dato viene tenuto segreto dalle aziende produttrici, possiamo affermare che un numero eccessivo di parametri non porta sempre ad un miglioramento nelle capacità di risposta di un sistema di Intelligenza Artificiale.

Alcune ricerche hanno dimostrato che raddoppiare il numero di parametri porta ad un miglioramento di pochi punti percentuali sulle capacità totali. Oltre a questa barriera vi è anche un aspetto economico: raddoppiare il numero di parametri spesso implica dover raddoppiare il quantitativo di hardware e di energia necessari.

1.4 I modelli più usati e alternative open-source

I modelli di Intelligenza Artificiale che sono al momento sul podio per il numero di utenti sono variegati. Non è possibile reperire dati certi per tale studio ma possiamo stabilire in modo empirico, tramite il sito Google Trends e l'osservazione diretta dell'attenzione online, i nomi dei sistemi di Intelligenza Artificiale più utilizzati e cercati.

Nella Figura 1.1 riportiamo un estratto dove sono stati individuati i 5 modelli di Intelligenza Artificiale più popolari tra gli utenti.

Di particolare interesse è la cadenza delle ricerche nei sistemi di Intelligenza Artificiale: vediamo che durante la settimana lavorativa abbiamo dei picchi di utilizzo, mentre nei giorni di sabato e domenica il loro utilizzo diminuisce significativamente.

I tre modelli di Intelligenza Artificiale più ricercati e famosi al momento sono:

- ChatGPT;
- Gemini;
- Claude.

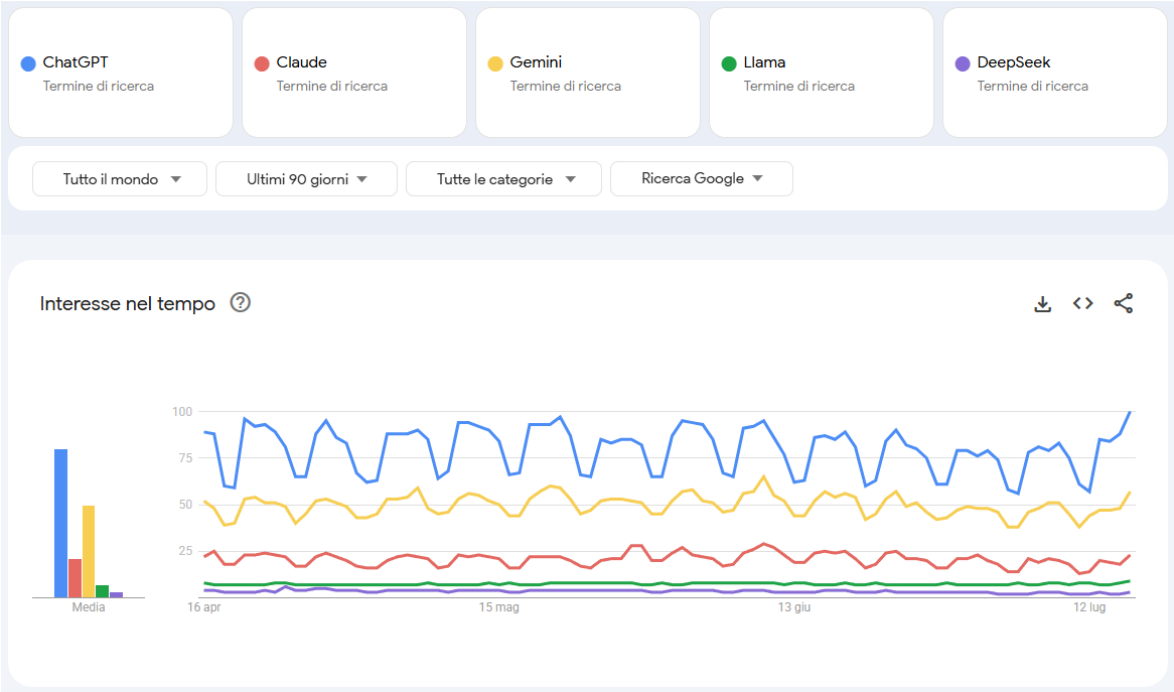


Figura 1.1: Grafico delle ricerche sui sistemi di Intelligenza Artificiale

Questi sono modelli chiusi appartenenti a compagnie private, non accessibili dall'esterno per un'analisi del loro codice sorgente. Come per altri aspetti del mondo che ci circonda, esistono alternative open-source ai modelli chiusi; essi possiedono innumerevoli vantaggi che i modelli chiusi non hanno. In particolare, essi possono essere personalizzati tramite la procedura di fine-tuning che permette di specializzarli su un argomento specifico e di personalizzare le risposte.

I tre modelli di Intelligenza Artificiale open-source più usati e famosi al momento sono:

- Llama;
- Mistral;
- DeepSeek.

Un'altro aspetto interessante dei modelli open-source è il fatto che possono essere installati in locale così da avere la propria Intelligenza Artificiale. Una persona che vuole essere assistita da un sistema di Intelligenza Artificiale ma non desidera immettere documenti importanti in rete può affidarsi a tecnologie locali. Se si caricassero i file personali in un sistema remoto, un possibile problema riguarderebbe la possibilità che le informazioni personali trapelino all'esterno.

La rete neurale dei sistemi di Intelligenza Artificiale leader al momento è molto vasta; gli stessi esperti non hanno mai visto ed esaminato tutti i nodi che la compongono.

Esistono testimonianze in rete in cui un modello di Intelligenza Artificiale manifesta dei comportamenti anomali che, nel momento dell'individuazione, vengono moderati dagli ingegneri con delle regole di filtraggio degli output del modello.

1.5 Campi di applicazione e impatto sociale

L'Intelligenza Artificiale Generativa è comparsa in un momento cruciale dello sviluppo tecnologico mondiale. Se prendiamo in esame la categoria mobile, al momento ci sono

molti più smartphone attivi che persone al mondo che li possano utilizzare. La tecnologia ci circonda e l'accesso ad un sistema di Intelligenza Artificiale può essere effettuato dalla maggior parte delle persone al mondo.

Questa tecnologia può essere applicata alla maggior parte dei settori umani. L'impatto maggiore vi è stato nel settore IT. Visto anche che gli esperti di Intelligenza Artificiale fanno parte di questo settore, sono stati i primi ad averla utilizzata.

Le applicazioni future riguarderanno tutte le situazioni in cui si opera con un computer con dei compiti ripetitivi.

L'implementazione di tale tecnologia possiede limiti di funzionamento di natura fisica: la carenza di semiconduttori, di generatori di energia elettrica a basso costo e di altre materie prime rappresenta una sfida nella ricerca e nello sviluppo. Tale scenario rende, però, possibili e convenienti le ottimizzazioni delle reti neurali per attivare il minor numero possibile di parametri, mantenendo, allo stesso tempo, invariata la qualità degli output generati.

Un aspetto che viene spesso trascurato è il cosiddetto "*Model Collapse*". Questo fenomeno è caratterizzato da un sistema di Intelligenza Artificiale che utilizza per il proprio addestramento dei dati che sono stati generati da se stesso o da altri sistemi di Intelligenza Artificiale. Il risultato è un collasso del modello in cui i dati vengono rimescolati e si vengono a creare condizioni in cui un sistema diventa instabile nella generazione di contenuto informativo coerente e corrispondente al vero.

Un ulteriore grande cambiamento è avvenuto nel mondo del coding. Al momento, tale settore ha implementato in modo massiccio l'uso di generatori di codice e meccanismi automatici di debugging. L'Intelligenza Artificiale ha sia coperto molto del lavoro ripetitivo che sostituito, almeno in parte, la creatività di ogni compito.

CAPITOLO 2

Descrizione degli LLM considerati

Il presente capitolo è dedicato all'analisi dettagliata di tre dei principali Large Language Model al momento disponibili nel mercato: OpenAI, con i modelli Codex e ChatGPT; Google, con Gemini; ed Anthropic con Claude.

La scelta di questi tre sistemi non è casuale; essi mantengono il maggior numero di utilizzatori del mondo occidentale, vista anche la vicinanza strategica tra i paesi utilizzatori e produttori.

Verranno analizzate le origini aziendali di ogni modello e si procederà con la spiegazione del loro contesto tecnologico.

Si definirà l'architettura e, ove possibile, le loro caratteristiche tecniche.

Se ci si sofferma all'osservazione dell'interfaccia utente di ogni modello considerato, si potrà notare una forte somiglianza. Questa scelta stilistica è una tecnica sociale ben documentata nel libro "The power of habit", in cui si cerca di riprodurre una user experience a noi famigliare con un prodotto o oggetto già esistente per invocare ricordi positivi. Procederemo, dunque, nell'analisi delle interfacce utente per ogni modello.

Verranno, infine, illustrate le modalità di input e le funzionalità integrate per ogni modello.

2.1 OpenAI - ChatGPT e l'ecosistema Codex

OpenAI rappresenta il punto di svolta nella storia recente degli LLM, visto che proprio questa azienda è riuscita a portare il prodotto dei sistemi di Intelligenza Artificiale al grande pubblico aggiudicandosi il primo posto anche per il brand.

ChatGPT sta all'AI come la Coca-Cola sta alla cola classica nella vita di tutti i giorni. Quando si cerca una risposta ad un determinato problema, si sente spesso la frase "chiedi a ChatGPT". Questo fenomeno si chiama *genericizzazione del marchio* e fornisce un netto vantaggio competitivo ad OpenAI.

2.1.1 Origini e contesto tecnologico

OpenAI è stata fondata nel 2015 a San Francisco USA, da Elon Musk, Sam Altman e altri investitori. Essa era nata come un laboratorio no-profit.

L'obiettivo iniziale era la progettazione di un'Intelligenza Artificiale Generale (AGI), di cui tutti gli esseri umani potessero beneficiare, supportata anche dal rilascio continuo e libero delle proprie ricerche.

L'addestramento dei modelli, che presto superarono dimensioni notevoli, portò costi rilevanti e, visto anche il crescente interesse nel settore, tale società divenne a scopo di lucro nel 2019.

2.1.1.1 ChatGPT

Un chatbot fu inizialmente rilasciato nel 2018 nella prima versione, GPT-1, e aggiornato successivamente nella versione GPT-3 nel 2020.

Per lo sviluppo di ChatGPT, OpenAI prese come esempio la tecnologia contenuta nel documento pubblicato da Google, "Attention Is All You Need", e la rese più sofisticata. ChatGPT ha una dimensione notevole; conta circa 175 miliardi di parametri. Inoltre, in GPT-3 è presente l'RLHF (Reinforcement Learning from Human Feedback), una tecnica che, grazie ad un feedback umano, apprende e modifica le proprie risposte per adattarsi al contesto, attuando un rinforzo delle proprie conoscenze.

Le versioni successive contengono un numero sempre maggiore di parametri totali.

OpenAI, non avendo un proprio canale ufficiale di distribuzione, cerca di aggregarsi ad aziende ben consolidate. Vediamo, in particolare, i successi maggiori:

- *Apple Intelligence*, dove OpenAI ha stretto una partnership strategica per integrare i propri sistemi all'interno di Siri, l'assistente ufficiale di Apple;
- *Windows e Office 365*, integrando nativamente il proprio sistema di Intelligenza Artificiale con Copilot;
- *app terze*, si riesce ad accedere a servizi terzi direttamente da ChatGPT.

2.1.1.2 Codex

Gli ambienti di sviluppo software sono nati da un'intuizione: il codice come linguaggio naturale. Sebbene la sintassi dei linguaggi di programmazione comuni sia rigida, sintatticamente e semanticamente i linguaggi seguono un filo logico predittivo proprio come un linguaggio naturale. Questo permette di predire la logica dei blocchi di codice successivi da comporre per uno script, conoscendo il blocco iniziale di codice.

I modelli di Intelligenza Artificiale come GPT-3 avevano un'infarinatura di base di generazione di codice, visto che per l'addestramento veniva utilizzato anche il materiale trovato sul web, che poteva contenere guide e frammenti di codice.

Tuttavia GPT-3 soffriva di tre problemi principali ovvero:

- *allucinazioni del codice*; GPT-3 non riusciva a generare codice coerente per risolvere un problema specifico spiegato attraverso il prompt;
- *errori sintattici e semantici*; nonostante GPT-3 generasse codice apparentemente corretto, tale codice causava errori durante la compilazione o l'interpretazione;
- *carenza di logica funzionante*; il modello, nel momento in cui riusciva a generare un codice sintatticamente corretto, non riusciva sempre a risolvere il problema originale.

Applicando a GPT-3 la tecnica del fine-tuning, gli sviluppatori di OpenAI sono riusciti a portare alla luce Codex. Per il suo sviluppo OpenAI ha usato un dataset di dimensioni notevoli, prelevando gran parte dei dati dai repository pubblici di GitHub, dove migliaia di programmatori pubblicano il codice da loro scritto.

Codex fu rilasciato nel 2021 e inizialmente non era sviluppato come una chat: era un motore backend accessibile tramite API.

La sua migliore applicazione è stata l'integrazione nativa in GitHub Copilot. Questo diede inizio alla fase del "Pair Programming", dove il modello divenne un suggeritore di programmazione integrato automaticamente in ambienti di sviluppo (IDE) come VS Code.

Nel 2023 OpenAI ha deprecato i modelli standalone di Codex basati su API, considerati ormai obsoleti.

Grazie allo sviluppo successivo di GPT-4, un modello che offre grandi capacità di ragionamento logico, Codex è stato integrato nell'ambiente di ChatGPT.

2.1.2 Architettura e caratteristiche tecniche

Essendo dei pionieri nello sviluppo di un sistema di Intelligenza Artificiale per il mondo consumer, non vi erano riferimenti funzionali su cui attuare degli studi di ottimizzazione. OpenAI scelse di sviluppare inizialmente modelli unimodali, per poi diventare progressivamente multimodali, vista anche la popolarità dei modelli dei loro competitor.

2.1.2.1 ChatGPT

Questo modello di Intelligenza Artificiale sfrutta la tecnologia Transformer. All'interno di un testo ci sono relazioni più o meno marcate tra parole; ChatGPT riesce ad analizzare tale testo e a individuare relazioni anche tra termini lontani tra loro.

All'inizio, ciò che ha distinto lo sviluppo di ChatGPT dagli altri modelli di Intelligenza Artificiale è la tecnologia RLHF (Reinforcement Learning from Human Feedback), come spiegato nel paragrafo precedente: grazie ai feedback umani il modello modifica e rinforza la qualità delle proprie risposte.

2.1.2.2 Codex

L'architettura di Codex non è più distinta da quella di ChatGPT, come abbiamo già spiegato; esso non opera più come una tecnologia backend tramite le vecchie API (che sono state deprecate), ma è stato assorbito e integrato direttamente nei modelli di ChatGPT.

Quando si utilizza un agente per il codice, questo opera generalmente all'interno di una sandbox, che è un ambiente isolato sulla cartella di progetto e, tramite prompt, gli si indica quali operazioni eseguire.

Un aspetto importante da tenere a mente è che, nel momento in cui si scrive un prompt, occorre specificare chiaramente dove effettuare le modifiche perché altrimenti, Codex potrebbe modificare parti del software che non si desiderano toccare.

L'applicazione desktop di Codex è stata rivoluzionata nel 2026 e integrata direttamente nell'applicazione di ChatGPT.

2.1.3 Interfaccia utente

L'interfaccia utente di questi due sistemi è molto simile tra loro, sono due chatbot, ma sono progettati per due campi di applicazione diversi, ovvero testo e software.

2.1.3.1 ChatGPT

In questo paragrafo vedremo gli aspetti grafici dell'interfaccia utente e le caratteristiche funzionali di ChatGPT.

Nella Figura 2.1 è rappresentata l'interfaccia utente di ChatGPT, la schermata presenta in posizione centrale un campo di testo in cui si inserisce il prompt da inviare per l'elaborazione del modello.

Sono, inoltre, presenti una serie di comandi:

- *il pulsante* + permette di aggiungere file dal proprio dispositivo e selezionare dei plugin;

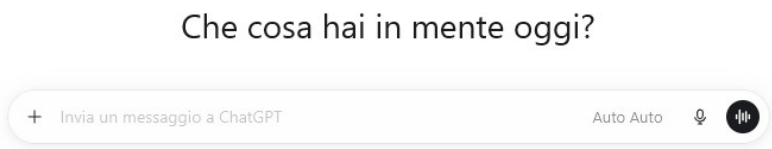


Figura 2.1: Interfaccia di ChatGPT

- *il pulsante "detta"* viene utilizzato per la dettatura vocale del testo;
- *il pulsante "voice"* è necessario per avviare una chat vocale con il modello di Intelligenza Artificiale.

2.1.3.2 Codex

In questo paragrafo vedremo gli aspetti grafici dell'interfaccia utente e le caratteristiche funzionali di Codex.

Nella Figura 2.2 è rappresentata l'interfaccia utente di Codex, la schermata presenta in posizione centrale un campo di testo in cui si inserisce il prompt da inviare per l'elaborazione da parte del modello.

Sono, inoltre, presenti una serie di comandi, ovvero:

- *il pulsante* +, che permette di aggiungere file dal proprio dispositivo e selezionare dei plugin;
- *il pulsante "detta"*, per la dettatura vocale del testo;
- *il pulsante "voice"*, necessario ad avviare una chat vocale con il modello di Intelligenza Artificiale;
- *il pulsante di profondità del ragionamento*, necessario per specificare al modello di pensare più a lungo per elaborare la risposta;
- *i pulsanti di suggerimento dei prompt*, che sono delle azioni rapide con esempi di richieste da cui iniziare.

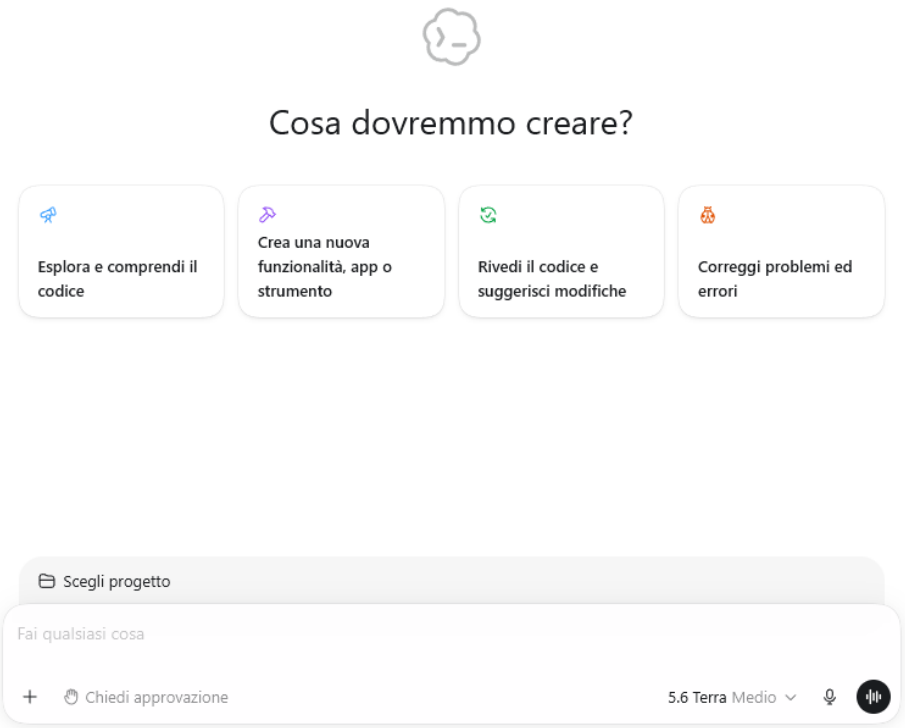


Figura 2.2: Interfaccia di Codex

2.1.4 Modalità di input e funzionalità integrate

I modelli di Intelligenza Artificiali, come ChatGPT e Codex, possono gestire molte modalità di input, consentendo l'analisi di molteplici tipologie di file. Inoltre, tali modelli possiedono diverse tipologie di funzionalità integrate che risultano particolarmente utili in molteplici contesti applicativi.

2.1.4.1 ChatGPT

L'applicazione di ChatGPT presenta un formato comune per i sistemi di Intelligenza Artificiale in quanto si utilizzano le classiche modalità di input. Inoltre, abbiamo delle funzionalità che possono essere utili durante l'utilizzo della chat. I principali metodi di input sono:

- *testuale*, in cui si scrive un prompt anche di grandi dimensioni; supporta sia le istruzioni iniziali del sistema (system prompt) sia formattazioni complesse, come JSON;
- *file multiformato*, per l'analisi di molte tipologie di file, come, ad esempio `PDF`, `TXT`, documenti `DOX` o immagini `PNG`.

Le funzionalità integrate sono:

- *navigazione web*; permette di accedere alle informazioni aggiornate della rete direttamente dai prompt, senza utilizzare un browser;
- *generazioni di immagini*; consente di creare immagini personalizzate a partire da un prompt testuale o rielaborando immagini.

2.1.4.2 Codex

L'ecosistema Codex presenta delle modalità di input simili agli altri modelli ma con una distinzione particolare: l'ambiente di lavoro è ottimizzato per un progetto. Le modalità di input sono:

- *descrizione in linguaggio naturale*; il modello è abilitato a ricevere specifiche e descrizioni funzionali di un software scritte in testo libero;
- *codice sorgente o interi progetti*; è possibile inserire snippet di codice o allegare file di un progetto per chiederne il debugging, la spiegazione o il refactoring.

Le funzionalità integrate sono:

- *ambiente di esecuzione sandbox*, in cui è possibile limitare l'accesso di Codex solo a particolari percorsi al progetto;
- *coding agentico*, in cui è possibile pianificare ed eseguire passaggi sequenziali autonomi sul codice, come, ad esempio, la correzione automatica degli errori.

2.2 Google - Gemini

Questa sezione è dedicata a Google e alla sua famiglia di modelli Gemini che è la risposta di questa Big Tech al mercato degli LLM. Di seguito analizzeremo tale ecosistema, caratterizzato da un'impostazione nativamente multimodale e da una profonda integrazione di servizi cloud, hardware e software proprietario.

2.2.1 Origini e contesto tecnologico

Nel 2017 fu proprio Google ad inventare l'architettura base dei Transformer per gli LLM, e tale innovazione fu pubblicata nel saggio denominato "Attention is all you need". Tuttavia, per sicurezze aziendali e politiche interne, inizialmente non fu rilasciato nulla sul mercato consumer.

Con il passare del tempo Google si trovò a competere con l'emergente OpenAI e, anche per questo motivo, attuò una fusione tra Google Brain e DeepMind fondando l'attuale Google DeepMind, con l'obiettivo principale di creare il nuovo modello Gemini.

Per lo sviluppo e il funzionamento di Gemini, Google possiede un enorme vantaggio rispetto ad altri competitor, avendo una struttura aziendale solida e integrata.

In particolare, Google presenta i seguenti vantaggi:

- *hardware proprietario e indipendenza*; Google sviluppa e utilizza TPU, chip di Intelligenza Artificiale realizzati direttamente da essa;
- *dati multimodali unici*; possedendo un vasto ecosistema di servizi che producono dati unici, Google può sfruttare tali dati direttamente per l'addestramento;
- *infrastruttura globale di data center*, Google garantisce una disponibilità computazionale distribuita a livello mondiale;
- *canali di distribuzione*; Google possiede canali di collegamento nativi agli utenti, come Android, per la diffusione dei propri prodotti e servizi.

2.2.2 Architettura e caratteristiche tecniche

Google, con il proprio modello di Intelligenza Artificiale Gemini, ha introdotto scelte architetturali differenti rispetto a OpenAI, concentrandosi particolarmente nello scalare le dimensioni della tecnologia e nella creazione di un modello standardizzato. In particolare evidenziamo le seguenti caratteristiche:

- *Finestra di contesto estesa*; le versioni precedenti di modelli di Intelligenza Artificiale Generativa erano in grado di gestire un quantitativo di token che andava da 8000 unità per i modelli più anziani, fino a 128000 unità per i modelli più moderni. Gemini 1.5 Pro è stato il primo a poter gestire un milione di token e fino a due milioni nella versione estesa. Questo ha dato al modello una capacità molto estesa di recupero delle informazioni da più documenti che corrispondono a circa 700.000 parole.
- *Architettura Mixture of Experts*; il modello possiede un elevato numero di parametri totali; l'architettura in questione permette l'attivazione di sottosezioni specializzate in un determinato contesto tecnologico, riducendo in modo significativo il costo computazionale e la latenza delle risposte generate.
- *Multimodalità nativa*; Google, fin dal momento dello sviluppo di Gemini, ha optato per una progettazione del proprio sistema di Intelligenza Artificiale nativamente multimodale. L'architettura di Gemini consente che file testuali, video e immagini possano essere elaborati in un unico spazio e le informazioni sviluppate siano correlate tra loro.

2.2.3 Interfaccia utente

In questo paragrafo vedremo gli aspetti grafici dell'interfaccia utente e le caratteristiche funzionali di Gemini.



Figura 2.3: Interfaccia di Gemini

Nella Figura 2.3 è rappresentata l'interfaccia utente di Gemini, la schermata presenta in posizione centrale un campo di testo in cui inserire il prompt da inviare per l'elaborazione del modello.

Sono, inoltre, presenti una serie di comandi, ovvero:

- *il pulsante* +; permette di aggiungere file dal proprio dispositivo, dal Drive e selezionare dei plugin;
- *il pulsante con l'icona del microfono*; per la dettatura vocale del testo;
- *modalità di ragionamento*; necessario per selezionare la tipologia di ragionamento per la risposta al prompt.

2.2.4 Modalità di input e funzionalità integrate

A differenza di altri sistemi di Intelligenza Artificiale, Gemini accetta nativamente diverse tipologie di input e file, rendendosi un modello molto versatile. Le funzionalità che offre tale modello sono uniche e rispettano pienamente l'ecosistema della società che l'ha sviluppato.

Le modalità di input di Gemini sono:

- *testuale*, prompt formati da testo;
- *file multiformato*, per l'analisi di molte tipologie di file, come documenti `PDF`, `video`, `audio` e `immagini`.

Le funzionalità integrate sono:

- *analisi delle immagini*; essa è una delle funzionalità principali del piano gratuito; permette di far analizzare velocemente al modello un'immagine o uno screenshot senza dover copiare testo e senza particolari limitazioni nel normale utilizzo;
- *integrazione nativa con l'ecosistema Google*; collegamento diretto con quasi la totalità dei sistemi Google;
- *Google Search*; integrazione con il motore di ricerca di Google con un pulsante "double-check" per la verifica delle fonti;
- *creazione di immagini e audio*; permette di creare immagini tramite prompt testuali.

2.3 Anthropic - Claude

L'ultimo sistema di Intelligenza Artificiale preso in analisi è sviluppato da Anthropic, azienda nata con una forte impronta sull'etica e sicurezza dei propri sistemi di Intelligenza Artificiale.

Le prossime sottosezioni illustrano le origini, le specifiche tecniche e le funzionalità della famiglia dei modelli Claude, volte a favorire l'integrazione con i grandi servizi esterni già presenti sul mercato.

2.3.1 Origini e contesto tecnologico

Claude è stato creato da Anthropic, una società fondata nel 2021 da un gruppo di ricercatori ed ex dipendenti di OpenAI. La separazione dei ricercatori dall'azienda di origine, a quanto dichiarato, è avvenuta per motivi strettamente etici, in quanto il modello di business di OpenAI prevedeva il rilascio continuo dei modelli senza revisioni di sicurezza approfondite.

La natura societaria di Anthropic è "Public Benefit Corporation" cioè la generazione di profitto per gli azionisti e un impatto positivo sulla società e nella sicurezza. Il sistema di Intelligenza Artificiale di Anthropic è stato progettato con lo scopo di essere "helpful, honest and harmless", ovvero utile, d'aiuto e inoffensivo.

I campi principali in cui è stato introdotto Claude di Anthropic sono:

- *Enterprise Legal and Finance*; grazie a forti capacità di lettura di testi particolarmente lunghi e alla riduzione delle allucinazioni, Claude ha avuto una forte integrazione in questi settori;
- *Software Engineering*; essendo fortemente specializzato sullo sviluppo software e sull'integrazione di API aziendali, Claude è diventato uno dei sistemi di Intelligenza Artificiale preferiti dagli sviluppatori;
- *Partnership strategiche*; Amazon AWS e Google Cloud hanno stretto con Anthropic alleanze importanti, sia per investimenti strategici che per l'integrazione del modello sulle proprie piattaforme.

2.3.2 Architettura e caratteristiche tecniche

Anthropic, con il proprio modello di Intelligenza Artificiale Claude, si focalizza su un approccio in cui la sicurezza e la coerenza logica sono il punto focale del suo sviluppo. L'architettura del modello si basa sui Transformer e sull'ottimizzazione del meccanismo di attenzione per poter gestire dei prompt molto lunghi e complessi grazie ad una finestra di contesto estesa.

I modelli di Anthropic non sono unici, ma sono stati suddivisi in categorie in base alle singole capacità, come costi e velocità di computazione. In particolare si hanno:

- *Sonnet*; è un modello con una combinazione velocità e intelligenza per attività quotidiane;
- *Opus*; è un modello per la programmazione agentica e il lavoro aziendale;
- *Haiku*; è il modello più veloce con ottima intelligenza ma con finestra di contesto ridotta rispetto agli altri;
- *Fable*; è il modello che viene descritto come il più capace tra tutti gli altri.

2.3.3 Interfaccia utente

In questo paragrafo vedremo gli aspetti grafici dell'interfaccia utente e le caratteristiche funzionali di Claude.

Nella Figura 2.4 è rappresentata l'interfaccia utente di Claude, la schermata presenta in posizione centrale un campo testuale in cui si inserisce il prompt da inviare per l'elaborazione del modello.

Sono, inoltre ,presenti una serie di comandi, ovvero:

- *il pulsante +*; che permette di aggiungere file dal proprio dispositivo e di selezionare dei plugin;
- *il pulsante usa il microfono*; per la dettatura vocale del testo;
- *il pulsante modello*; necessario per selezionare il modello desiderato, ciascuno dei quali è specializzato in ambiti diversi.

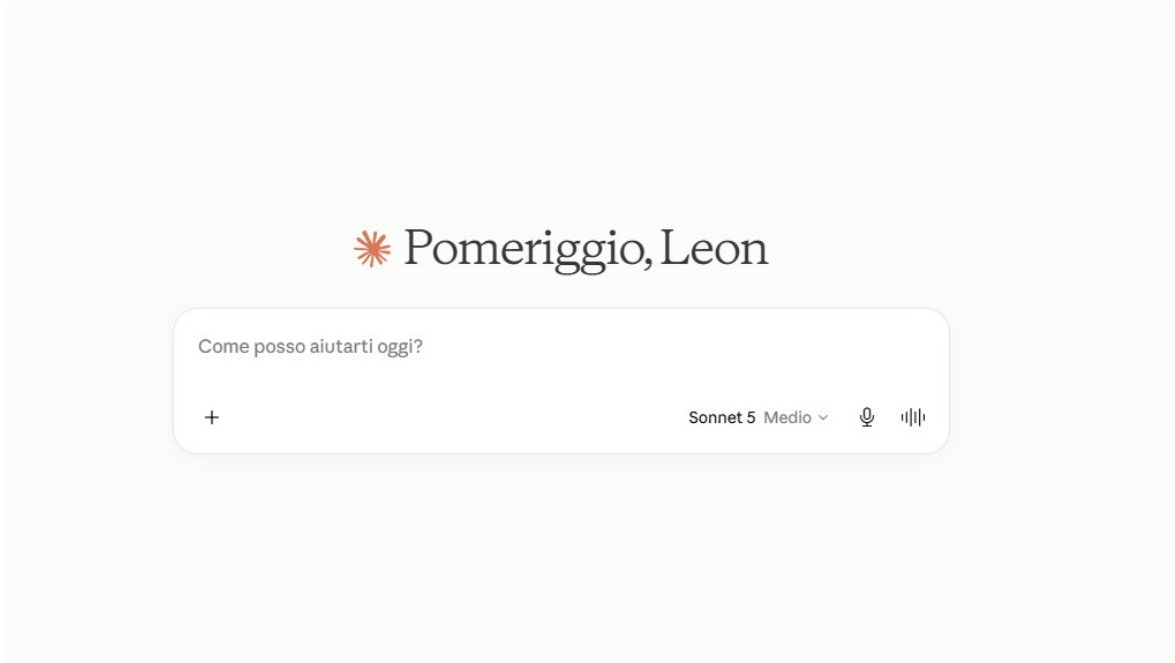


Figura 2.4: Interfaccia di Claude

2.3.4 Modalità di input e funzionalità integrate

Come per altri sistemi di Intelligenza Artificiale, Claude possiede molte modalità di input e funzionalità integrate che permettono di lavorare in maniera più agile su problemi e progetti complessi.

Le modalità di input di Claude sono:

- *testo e documenti complessi*; permette un'analisi di una grande quantità di documenti testuali;
- *Claude Vision*; permette l'analisi grafica di immagini o tabelle.

Le funzionalità integrate sono:

- *Artifacts*; permette di aprire una finestra laterale alla chat in cui è possibile visualizzare e lavorare in tempo reale a codice e file.
- *Basi di conoscenza*; in cui è possibile caricare file propri che verranno usati successivamente dal modello per la creazione di risposte personalizzate.
- *Tool use*; che permette di utilizzare funzionalità e API esterne nella propria chat.

2.3.5 Analisi comparativa dei modelli considerati

Diventa sempre più complesso orientarsi tra i sistemi di Intelligenza Artificiale tuttavia, possiamo individuare i punti di forza e dare una valutazione ai modelli considerati in base a come vengono utilizzati maggiormente. Tale risultato permette di valutare quale sistema di Intelligenza Artificiale è più adatto per un determinato compito e quale è da evitare.

La tabella 2.1 fornisce uno schema delle principali caratteristiche dei tre modelli.

	ChatGPT/Codex	**Gemini**	**Claude**
Sviluppato da	OpenAI	Google	Anthropic
Caso d'uso ideale	Produttività generale	Integrazione Google	Scrittura e codice
Integrazioni di sistema	Microsoft / app terze	Google WorkSpace	Strumenti per sviluppatori
Generazione di testo	ottima	buona	eccellente
Generazione di codice	eccellente	buona	eccellente
Generazione di immagini	eccellente	eccellente	assente

Tabella 2.1: Capacità e punti di forza dei sistemi di Intelligenza Artificiale considerati

CAPITOLO 3

Confronto degli LLM nella gestione dei testi

Nei capitoli precedenti è stata offerta una panoramica sull'Intelligenza Artificiale Generativa, partendo dalle origini per poi ripercorrere la storia contemporanea. Sono stati descritti i principali modelli utilizzati al momento, evidenziando i punti di forza che li rendono validi assistenti e i campi di applicazione in cui si sono consolidati con successo. In questo capitolo sposteremo il focus sul contesto pratico attraverso un case study: la valutazione di due modelli di Intelligenza Artificiale Generativa, ChatGPT e Gemini, per testarne le capacità analitiche. L'obiettivo atteso è la guida dell'utente verso la soluzione di un problema concreto.

3.1 Introduzione e definizione del problema

In questa sezione, verrà presentata una panoramica generale del problema da risolvere. Verranno, inoltre, introdotti i criteri di valutazione con cui si eseguirà la validazione della soluzione generata dai due sistemi di Intelligenza Artificiale.

3.1.1 Descrizione del problema testuale

Il case study riguarda un'automobile che presenta un malfunzionamento visibile, udibile e rilevabile tramite il codice errore mediante uno scanner OBD-2. Possiamo elencare la seguente sintomatologia dell'automobile difettosa:

- *vibrazione anomala del motore*, presente al minimo dei giri (RPM); è caratterizzata da un periodo temporale irregolare tra la fase in cui il motore funziona in modo fluido e quella in cui sono presenti le vibrazioni;
- *scarico scoppiettante*; produce dei colpi secchi con assenza di fumo visibile;
- *irregolarità di accelerazione*; si manifesta nel momento in cui avviene la pressione totale dell'acceleratore;
- *potenza ridotta*; l'automobile presenta un netto calo delle prestazioni generali.

3.1.2 Obiettivo della sperimentazione

Molti dei sistemi di Intelligenza Artificiale sono nativamente multimodali; tale caratteristica intrinseca ne permette l'utilizzo in molteplici scenari applicativi.

Nella nostra sperimentazione, metteremo alla prova questi sistemi per diagnosticare, grazie alla loro capacità di ragionamento ed elaborazione dei dati, un guasto meccanico o elettrico di un'automobile.

3.1.3 Criteri di valutazione

Indeterminatamente dal risultato della sperimentazione, occorre fissare dei criteri di valutazione per comprendere se l'interazione con l'utente sia stata profittevole o meno. Durante l'analisi comparativa e la discussione finale, utilizzeremo i dati ricavati per trarre le conclusioni.

I criteri di valutazione qualitativa dell'interazione con il sistema di Intelligenza Artificiale sono:

- *pertinenza al contesto*; valuta se il modello sia riuscito a generare una risposta pertinente al problema o se abbia risposto in maniera generica;
- *presenza di errori*; permette di valutare se il modello abbia fatto degli errori concettuali o di calcolo;
- *deriva conversazionale*; valuta se il modello abbia portato l'utente in una direzione diversa dalla soluzione;
- *soluzione al problema*; è il parametro più importante; valuta se il modello sia riuscito, anche dopo diverse interazioni con l'utente, a suggerire una soluzione o a risolvere il problema.

Oltre alla valutazione qualitativa, è necessario procedere con una valutazione quantitativa; elenchiamo, quindi, i criteri per quest'ultima:

- *numero di interazioni*; indica quanti prompt totali sono stati necessari per giungere alla diagnosi corretta;
- *riformulazioni del prompt*; indica il numero di volte in cui il modello ha necessitato di una riscrittura del prompt, in quanto non ne aveva compreso il contenuto;
- *customer satisfaction score*; indica quanto l'utente sia soddisfatto dell'interazione con il modello;
- *customer effort score*; indica quanto si stato facile risolvere il problema con l'aiuto del modello.

3.2 Metodologia di analisi

In questa sezione definiremo i modelli di Intelligenza Artificiale selezionati per il nostro test; definiremo, anche, delle linee guida, sia per la formulazione del prompt iniziale, che per quelli secondari.

3.2.1 Selezione dei modelli di Intelligenza Artificiale

Per eseguire la sperimentazione abbiamo scelto di testare due sistemi di Intelligenza Artificiale: Gemini e ChatGPT. Le caratteristiche principali che hanno guidato la scelta di tali modelli sono:

- *ampia finestra di contesto*; permette al modello di mantenere in memoria un'elevata quantità di informazioni della stessa sessione;
- *elevato numero di parametri*; consente di esplorare numerose possibilità e di accedere a un vasto bagaglio di conoscenze;
- *diversità di ragionamento*; permette di selezionare il livello di ragionamento a seconda della complessità del compito richiesto.

3.2.2 Progettazione del prompt iniziale

Sulla base della descrizione testuale del problema riportata nella Sezione 3.1, si definisce il prompt iniziale che verrà inviato al sistema di Intelligenza Artificiale Generativa.

Utente

Kia Sportage del 2008 benzina/GPL, fino a poco tempo fa quando percorrevo la strada in salita e premevo l'acceleratore fino in fondo, il motore saltava qualche colpo nei cilindri. Ora anche al minimo, sia a benzina che a GPL, il motore alterna il funzionamento tra 4 cilindri e 3 cilindri, mentre in salita, se non tengo il motore accelerato, funziona a 3 cilindri. Ho il codice errore, P0301, leggendolo sul dispositivo collegato alla porta OBD-2. Quale potrebbe essere il problema e quali controlli potrei eseguire?

Notiamo come il prompt iniziale risulti essere specifico nei dettagli tecnici, che sono stati considerati rilevanti per la diagnosi finale, in cui si chiede all'Intelligenza Artificiale di suggerire le possibili cause e vari test da effettuare.

3.2.3 Vincoli progettuali dei prompt secondari

Poiché i prompt secondari dipendono interamente dalla risposta fornita al prompt primario, la loro progettazione si limiterà a seguire un filo logico e naturale rispetto alle interazioni precedenti.

Un aspetto fondamentale riguarda il bilanciamento del livello di dettaglio: i prompt secondari devono essere specifici o generali nei punti giusti. Questo permette al modello sia di approfondire gli aspetti tecnici suggeriti, sia di conservare una sufficiente libertà di ragionamento. Esiste, infatti, un fenomeno noto come *prompt bias*, per cui un contesto o una formulazione troppo orientati da parte dell'utente possono condizionare negativamente le risposte del modello. Diventa inevitabile che le soluzioni proposte dal sistema risultino errate o fuorvianti nel contenuto.

3.3 Esecuzione del test

In questa sezione procederemo con la sperimentazione di due modelli di Intelligenza Artificiale per il case study precedentemente definito, applicando ad entrambi lo stesso prompt iniziale.

3.3.1 Scenario di test con Gemini

Illustriamo di seguito il risultato generato interrogando il sistema di Intelligenza Artificiale Generativa Gemini con il prompt iniziale:

Gemini

Il codice di errore P0301 indica in modo specifico un mancato avviamento (misfire) nel cilindro 1.
Il fatto che inizialmente il problema si presentasse solo sotto sforzo (in salita con acceleratore a tavoletta) e che ora si verifichi anche al minimo, sia a benzina che a GPL, restringono notevolmente il campo. Quando un problema affetta entrambi i carburanti, la causa è quasi sempre da ricercarsi nel sistema di accensione o, nel peggiore dei casi, nella meccanica del motore (compressione).
Ecco un'analisi dettagliata delle cause più probabili, ordinate dalla più semplice ed economica alla più complessa.
1. Il Sistema di Accensione (I principali indiziati)
Nella Kia Sportage del 2008, il sistema di accensione è fortemente sollecitato, soprattutto dal GPL che richiede una tensione di accensione più elevata rispetto alla benzina.

- *Le Candele di accensione*; Se non vengono cambiate da tempo, l'elettrodo si usura e la distanza aumenta. Questo spiega perché inizialmente il problema si presentasse solo in salita (quando la pressione nel cilindro aumenta e serve più energia per far scoccare la scintilla);
- *I Cavi candela*; Con il calore e il tempo, l'isolamento dei cavi si degrada. La corrente può "scaricare" a massa sul blocco motore invece di arrivare alla candela del cilindro 1;
- *La Bobina di accensione*; La Sportage di quell'anno utilizza spesso un sistema a bobine gemellate o una bobina singola con cavi. Se la porzione di bobina che gestisce il cilindro 1 si è fessurata o surriscaldata, smette di funzionare correttamente, lasciandoti a 3 cilindri.

2. Gioco Valvole e Usura Sedi Valvole (Criticità del GPL)
I motori Kia/Hyundai di quella generazione (soprattutto i 2.0 benzina) non hanno le punterie idrauliche autorodanti, ma richiedono la regolazione manuale del gioco valvole tramite spessori (piattelli).
Il GPL brucia a temperature più elevate rispetto alla benzina, causando la progressiva "recessione delle sedi valvole". Questo porta le valvole di scarico a "puntarsi" (rimanere leggermente aperte).
Perché combacia con i tuoi sintomi? Quando il motore è al minimo o sotto sforzo, una valvola che non chiude ermeticamente fa perdere compressione al cilindro 1, che smette di funzionare (andando a 3 cilindri). Quando acceleri, il tempo di perdita si riduce e il cilindro può riprendere a girare temporaneamente.
3. Iniettori (Meno probabile, ma da non escludere)
È raro che si guastino contemporaneamente l'iniettore benzina e l'iniettore GPL dello stesso cilindro. Tuttavia, se l'errore fosse stato presente solo a GPL, l'iniettore GPL del cilindro 1 sarebbe stato il colpevole perfetto (molto comune). Dato che lo fa a entrambi i carburanti, questa ipotesi passa in secondo piano.
Cosa fare adesso? (I passi per la diagnosi)
Per evitare di spendere soldi cambiando pezzi a caso, puoi fare (o far fare al meccanico) queste verifiche mirate:
Step 1: Il test d'inversione (Fai da te economico)

- *Invertire la candela*; Inverti la candela del cilindro 1 con quella del cilindro 2;
- *Invertire il cavo o la bobina*; Inverti il cavo candela (se la lunghezza lo permette) o la bobina del cilindro 1 con un altro cilindro;
- *Resettare l'errore*; Cancella l'errore OBD e guida l'auto finché non si riaccende la spia;
- *Se l'errore diventa P0302 (cilindro 2)*; Il problema è la candela o il cavo che hai spostato;
- *Se l'errore rimane P0301*; Il problema non è l'accensione esterna.

Step 2: Controllo Compressione e Gioco Valvole (Dal meccanico)
Se l'accensione è a posto, è fondamentale andare in officina e chiedere:

- *Misurazione della compressione*; Misurazione della compressione dei cilindri (per verificare lo stato del cilindro 1);
- *Verifica del gioco valvole*; Se l'auto non ha mai subito questa registrazione ed è sgasata da molti chilometri, è molto probabile che le valvole del cilindro 1 siano ormai prive di gioco.

Consiglio urgente: Evita di viaggiare a lungo con l'auto che va a 3 cilindri. La benzina o il GPL non bruciati nel cilindro 1 finiscono nel condotto di scarico, rischiando di distruggere il catalizzatore per via delle temperature altissime che si sviluppano lì dentro, trasformando una riparazione semplice in una molto costosa.

La risposta fornita dall'AI è estremamente accurata, ben formata e concettualmente corretta. Mostra una comprensione profonda del funzionamento dei motori bifuel, fornisce un'ottima guida iniziale per la verifica preliminare con consigli, sia per svolgere il lavoro in autonomia, sia per rivolgersi a un meccanico specializzato. L'unico errore notato è relativo alla grammaticalità del termine bobina, in cui l'AI lo scrive come "bobbina", dovuto, probabilmente, all'addestramento su forum di meccanica online.

L'utente effettua una parte delle verifiche e presenta al sistema il secondo prompt.

Utente

Ho appena acquistato e installato delle candele nuove e scambiato le due bobine, ma l'errore rimane P0301. Non ho potuto invertire i cavi perché il cilindro 1 ha il cavo più lungo di tutti.

Di seguito il risultato:

Gemini

Avendo già sostituito le candele e invertito le bobine senza successo (il codice resta P0301), hai escluso i due componenti d'accensione più comuni.
Tuttavia, c'è un dettaglio fondamentale sulla conformazione del motore 2.0 benzina (motore Beta II) della tua Kia Sportage 2008 che dobbiamo considerare prima di passare alla meccanica pesante:
Su questo motore, le due bobine sono montate direttamente sopra due cilindri (di solito il 2 e il 4), e da ciascuna bobina parte un cavo candela ad alta tensione che va ad alimentare il cilindro accoppiato (il cilindro 1 riceve la corrente tramite cavo dalla bobina sita sul cilindro 4).
Dato che hai cambiato le candele e invertito le bobine, i sospettati rimasti si stringono drasticamente. Ecco cosa resta da controllare, in ordine di probabilità:
1. Il Cavo Candela del Cilindro 1 (Ancora molto probabile)

- *stato del cavo*; anche se non hai potuto invertirlo perché è il più lungo, il cavo stesso è uno dei principali indiziati;
- *usura dell'isolamento*; con il calore del motore e gli anni, la gomma del cavo si logora e subisce delle microfratture (spesso invisibili a occhio nudo);
- *dispersione di corrente*; la corrente elettrica, invece di scendere fino alla candela, "salta" e scarica sul metallo del motore (la testata). Questo fenomeno peggiora drasticamente al minimo e sotto sforzo;
- *costo della riparazione*; un kit di cavi candela nuovi per la tua Sportage costa generalmente tra i 20 e i 40 euro ed è un lavoro da 5 minuti.

2. Il Gioco Valvole / Sedi Valvole Usurate (Probabilità Alta)

- *usura della meccanica*; se cambiando il cavo candela non risolvi, purtroppo entriamo nel campo della meccanica del motore, una situazione tipica dei motori a GPL di quell'epoca che hanno percorso molti chilometri;

- *regolazione e sedi valvole*; come accennato prima, questo motore ha la regolazione meccanica del gioco valvole. Con l'uso a GPL, le valvole di scarico tendono a "mangiarsi" la sede e a risalire, azzerando il gioco;
- *perdita di compressione*; quando il gioco si azzera, la valvola del cilindro 1 non chiude più perfettamente. Al minimo o in salita, la compressione scappa da quella valvola, il cilindro non fa scoppio e la centralina rileva il P0301;
- *azione consigliata*; serve un meccanico per effettuare un test di compressione sui cilindri. Se il cilindro 1 mostra una compressione significativamente più bassa degli altri, il problema sono le valvole puntate.

3. Cablaggio o comando della Bobina (Meno probabile)
Se il segnale elettrico che arriva dalla centralina alla bobina che gestisce il cilindro 1 ha uno spinotto ossidato o un filo rovinato, la scintilla sarà debole o assente.
Il prossimo passo consigliato
Prima di andare dal meccanico e spendere per test più approfonditi, ti consiglio di concentrarti sul cavo candela del cilindro 1.
Un metodo "artigianale" ma efficace per verificare se il cavo scarica a massa è questo: stasera, al buio pesto, accendi l'auto, apri il cofano e guarda attentamente la zona del cavo del cilindro 1 (soprattutto dove si infila nel motore). Se vedi delle minuscole scintille blu o avverti un ticchettio regolare, hai trovato il colpevole: il cavo è da buttare.
Se visivamente non vedi nulla, l'acquisto di un set di cavi nuovi è comunque il passo successivo più logico ed economico prima di procedere con il test di compressione in officina.

Gemini risponde in modo coerente e presenta una logica diagnostica impeccabile. Avendo già escluso due componenti elettrici del problema, il modello esclude un componente meccanico e presenta una nuova lista prioritaria. L'utente decide, quindi di presentare una nuova informazione con l'obiettivo di determinare se Gemini confermerà la propria tesi, oppure fornirà una nuova lista di possibili cause.

Utente

Ti fornisco un'altra informazione: se percorro la strada in pianura e tengo il motore da 2500giri/min in su, funziona bene.

Di seguito il risultato:

Gemini

Questa informazione è fondamentale e rappresenta il "pezzo del puzzle" mancante. Il fatto che sopra i 2500 giri/min in pianura il motore giri rotondo e funzioni bene esclude quasi del tutto un guasto elettrico totale (come una bobina o un cavo completamente interrotti) e stringe il cerchio su due sole opzioni.
Ecco perché il motore si comporta così e cosa sta succedendo:
I due unici sospettati rimasti
1. Valvole puntate / Perdita di compressione (Probabilità: 80%)
Nei motori a GPL, questo è il comportamento da manuale quando le valvole (solitamente quelle di scarico) hanno perso il gioco e non chiudono più ermeticamente.

- *funzionamento a 2500 giri*; a regimi elevati, i pistoni muovono i gas così velocemente che il tempo materiale in cui la valvola difettosa rimane "socchiusa" è infinitesimale. La perdita di pressione non fa in tempo a influenzare lo scoppio. Il cilindro 1 riesce a generare potenza;
- *comportamento al minimo o in salita*; al minimo, il motore gira piano; la compressione ha tutto il tempo di "scappare" dalla valvola che perde, annullando lo scoppio (P0301). In

salita a bassi giri, richiedi massima pressione nei cilindri (coppia), e la minima perdita fa cedere il cilindro.

2. Dispersione di corrente dal Cavo Candela (Probabilità: 20%)
Un cavo candela vecchio o lesionato fa fatica a far passare la corrente quando la pressione nel cilindro è alta (al minimo o sotto sforzo in salita).

- *funzionamento a 2500 giri*; a quel regime e in pianura, l'alternatore carica al massimo della tensione, e lo sforzo richiesto al motore è minimo. La scintilla riesce a vincere la resistenza e a scoccare nel cilindro invece di disperdersi.

Verdetto e cosa fare adesso
La combinazione dei tuoi sintomi (GPL, errore fisso sul cilindro 1, candele/bobine nuove, peggioramento sotto sforzo/minimo, funzionamento normale agli alti giri) punta dritto a un problema di compressione meccanica dovuto alle valvole.
Il passo successivo non è più sostituire pezzi a caso, ma fare una diagnosi mirata:

- *il test della notte (Fai-da-te)*; come ti dicevo prima, dai un'occhiata al cavo candela al buio pesto per escludere la dispersione elettrica. Se vedi scintille, cambia i cavi;
- *il test di compressione (Dal meccanico - Fondamentale)*; se i cavi sono a posto, vai dal meccanico e chiedi un test di compressione a freddo e a caldo (ci vogliono 15 minuti). Se il cilindro 1 mostra valori più bassi degli altri, la diagnosi è confermata: le valvole del cilindro 1 sono puntate.

Se il problema sono le valvole e l'hai preso in tempo, spesso basta effettuare la regolazione del gioco valvole (sostituendo i piattelli calibrati) per ridare vita al motore senza dover smontare la testata. Non aspettare troppo, perché se continui a usarla così rischi di bruciare letteralmente la valvola, richiedendo poi un intervento molto più costoso.

Gemini presenta una lista in cui indica due possibili cause al problema stimandone una al 20% di probabilità, l'altra all'80%. L'utente, valutando il costo di entrambi gli interventi, decide di scegliere il più economico, ma anche più improbabile.

Utente

Consultando il manuale di riparazione, nella sezione relativa alla guida per misurazione della resistenza elettrica dei cavi di accensione delle candele, è stata rilevata questa informazione: "la resistenza dei cavi non dovrebbe essere più alta di 5.6 kΩ per metro di cavo con scarto di +-20%. Se superiore a 10 kΩ per metro bisogna sostituire i cavi". Ebbene, sul cavo lungo 68cm del cilindro numero 1, ho una resistenza totale pari 9.6 kΩ.

Di seguito il risultato:

Gemini

Hai fatto benissimo a consultare il manuale e a usare il tester: i numeri che hai trovato ci danno una risposta matematica e inequivocabile.
Il cavo del tuo cilindro 1 è da sostituire immediatamente ed è la causa (o quantomeno la causa principale) del tuo problema.
Facciamo insieme il calcolo basandoci sui dati tecnici del manuale per capire perché il tuo valore è totalmente fuori specifica:
Il Calcolo Matematico sul tuo Cavo
Il manuale dice che il limite massimo tollerabile è 5.6 kΩ per metro (con un + 20%, arriviamo a un massimo assoluto di circa 6.7 kΩ per metro). Se supera i 10 kΩ per metro, il cavo è da buttare.
Il tuo cavo è lungo 68 cm (ovvero 0.68 metri). Calcoliamo quanta resistenza dovrebbe avere

proporzionalmente alla sua lunghezza:

- *valore nominale corretto*; 3.8 k Ω;
- *valore massimo con il +20%*; 4.57 k Ω;
- *soglia di sostituzione obbligatoria (i* 10 k Ω*/m)*; 6.8 kΩ;
- *valore misurato sul tuo cavo*; misura ben 9.6 k Ω.

Se rapportiamo il tuo valore su base metrica (6.8kΩ), il tuo cavo sta viaggiando a ben 14.1 kΩ per metro, ovvero ben oltre la soglia limite di 10 kΩ indicata dal manuale.
Perché questo spiega perfettamente i tuoi sintomi?
La resistenza interna del cavo del cilindro 1 è quasi il triplo di quanto dovrebbe essere.

- *al minimo e in salita (Motore a 3 cilindri)*; quando il motore è al minimo o sotto sforzo (salita), la pressione nella camera di scoppio è alta e l'aria è densa. In queste condizioni, serve una tensione elettrica altissima per far scoccare la scintilla sulla candela. Poiché il cavo offre una resistenza enorme (9.6 kΩ), la corrente "fatica" a passare e preferisce non scoccare affatto, oppure disperdersi verso il blocco motore. Il cilindro 1 si spegne;
- *sopra i 2500 giri in pianura (Il motore va bene)*; a quel regime la tensione generata dall'alternatore è massima e costante. Inoltre, in pianura a giri costanti la pressione interna del cilindro è minore rispetto a quando sei in salita. La corrente riesce a "sfondare" il muro di resistenza dei 9.6 kΩ e ad arrivare alla candela, facendo funzionare il cilindro.

Cosa fare ora
La diagnosi è confermata. Compra un kit di cavi candela nuovi.

- *consiglio sulla sostituzione*; cambiali tutti e quattro insieme. Se il cavo 1 è ridotto così, anche gli altri avranno una resistenza alterata dall'invecchiamento e presto ti darebbero lo stesso problema sui cilindri 2, 3 o 4. Il costo del kit completo è fortunatamente contenuto (solitamente tra i 20 e i 40 euro);
- *procedura finale*; una volta montati i cavi nuovi, cancella l'errore P0301 tramite OBD e fai una prova in salita. Con ottima probabilità avrai risolto il problema senza dover toccare il motore meccanicamente.

Gemini, con l'ultima informazione a lui fornita, consiglia di procedere con la sostituzione dei cavi delle candele. Tutto nasce dall'intuizione dell'utente nell'effettuare ricerche approfondite sul problema dei cavi e lasciare in secondo piano il problema più probabile. Prima della scrittura di questo prompt, l'utente ha utilizzato una funzione integrata in Gemini che permette di creare un branch secondario, valutare altre soluzioni e poi tornare al branch principale. Interrogando l'Intelligenza Artificiale nel branch secondario, sulla possibilità di continuare con la diagnostica sui cavi, il modello ha risposto negativamente. Solo grazie all'intuito umano è stato possibile indirizzarsi verso la soluzione al problema. Per confermare ulteriormente la tesi, l'utente ha misurato le resistenze di tutti i cavi e li ha riportati nel prompt successivo.

Utente

Ho effettuato la misurazione di tutti i cavi delle candele e ti inserisco i valori di resistenza in kΩ e la lunghezza in cm:

- c1 : 9.7 k Ω 68cm
- c2 : 7.1 k Ω 56cm
- c3 : 6.8 k Ω 47cm
- c4 : 5 k Ω 39cm

Quantifica in percentuale la probabilità che sia proprio questo il problema.

Di seguito il risultato:

Gemini

I dati che hai raccolto con il tester confermano in modo matematico che l'intero set di cavi è completamente degradato.
Se quantifichiamo la probabilità che il tuo problema (errore P0301 e passaggio a 3 cilindri al minimo/salita) sia causato esclusivamente o prevalentemente da questi cavi, la percentuale è del 90%.
Il restante 10% di dubbio rimane legato a una possibile concausa meccanica (il gioco valvole di cui parlavamo prima), ma l'evidenza elettrica qui è schiacciante.
Ecco l'analisi percentuale e matematica dei tuoi dati basata sul limite del manuale (5.6 kΩ per metro, massimo 6.7 kΩ/m considerando il +20% di tolleranza):

- *C1 (Errore P0301)*; lunghezza 0.68 m, resistenza rilevata 9.7 kΩ, limite massimo tollerabile 4.5 kΩ, fuori specifica del +112% (più del doppio);
- *C2*; lunghezza 0.56 m, resistenza rilevata 7.1 kΩ, limite massimo tollerabile 3.7 kΩ, fuori specifica del +88%;
- *C3*; lunghezza 0.47 m, resistenza rilevata 6.8 kΩ, limite massimo tollerabile 3.15 kΩ, fuori specifica del +115%;
- *C4*; lunghezza 0.39 m, resistenza rilevata 5 kΩ, limite massimo tollerabile 2.6 kΩ, fuori specifica del +90%.

Come puoi notare, tutti e quattro i cavi sono gravemente fuori specifica, con valori che surclassano il limite massimo del manuale.
Perché il Cilindro 1 ha ceduto per primo?

- *analisi chilometrica*; guardando i dati reali rapportati al metro, la situazione del cilindro 1 e del cilindro 3 è la peggiore in assoluto (viaggiano entrambi a oltre 14 kΩ per metro contro il limite di sbarramento di 10 kΩ/m);
- *lunghezza del cavo*; il cilindro 1 è quello che ha ceduto per primo perché, essendo il cavo fisicamente più lungo (68 cm), accumula la resistenza totale più alta in assoluto (9.7 kΩ);
- *comportamento elettrico*; per la corrente elettrica, quel cavo è diventato un muro quasi invalicabile al minimo e sotto sforzo. Il cilindro 3 sarebbe stato sicuramente il prossimo a spegnersi e a generare un errore P0303.

Conclusione sulla probabilità (90%)
La diagnosi è praticamente certa. Quando la resistenza è così alta, la bobina non riesce a spingere abbastanza tensione fino alla candela. La scintilla semplicemente non avviene dentro al cilindro, oppure la corrente preferisce "bucare" l'isolamento esterno del vecchio cavo e scaricarsi sul blocco motore.
Cosa fare ora:

- *acquisto dei ricambi*; procedi a colpo sicuro con l'acquisto di un kit di cavi candela nuovi di buona qualità (es. NGK, Bosch, Magneti Marelli);
- *ripristino della vettura*; una volta installati e azzerato l'errore dalla centralina, la tua Sportage dovrebbe tornare a girare a 4 cilindri in qualsiasi condizione.

L'utente acquistando, misurando e sostituendo i cavi dell'automobile, giunge alla soluzione del problema.

Utente

Ho sostituito i cavi delle candele e ho effettuato la misurazione dei cavi nuovi:

- c1 : 3.8 k Ω 68cm
- c2 : 3.1 k Ω 56cm
- c3 : 2.6 k Ω 47cm
- c4 : 2 k Ω 39cm

Dopo la sostituzione dei cavi il problema del motore è stato risolto.

Di seguito il risultato:

Gemini

Grandioso! I numeri dei cavi nuovi sono semplicemente perfetti: se guardi i valori (3.8 kΩ per il cavo da 68 cm), corrispondono esattamente al valore nominale da manuale (3.8 kΩ).
Hai fatto un lavoro di diagnosi da manuale. Invece di andare alla cieca cambiando pezzi a caso, hai usato la logica, il tester e i dati tecnici, trovando il colpevole al 100%.
Ora che la corrente arriva forte e costante a tutte e quattro le candele, la bobina non deve più "faticare" per superare quel muro di resistenza che c'era prima.
I vantaggi del lavoro svolto:

- *risoluzione del guasto*; risolto definitivamente il problema del motore che andava a 3 cilindri;
- *protezione delle bobine*; le bobine nuove non rischiano più di bruciarsi a causa dell'eccessiva resistenza elettrica;
- *salvaguardia del catalizzatore*; evitato il danneggiamento del catalizzatore di scarico causato dal carburante incombusto.

Goditi la tua Kia Sportage di nuovo a 4 cilindri fissi e complimenti ancora per l'ottima riparazione fai-da-te!

3.3.2 Scenario di test con ChatGPT

Illustriamo di seguito il risultato generato interrogando il sistema di Intelligenza Artificiale Generativa ChatGPT con il prompt iniziale:

ChatGPT

Il codice P0301 indica un mancato scoppio (misfire) del cilindro n. 1. Il fatto che il problema:

- *carburante*; si presenti sia a benzina che a GPL;
- *evoluzione*; sia peggiorato nel tempo;
- *condizioni*; ora sia presente anche al minimo;

fa pensare che il problema non sia nell'impianto GPL, ma in un componente comune ai due sistemi.
Le cause più probabili, in ordine di probabilità, sono:

- *candela del cilindro 1*; la causa più semplice ed economica. Se è usurata, sporca o con elettrodo consumato, provoca esattamente questi sintomi. Se le candele hanno più di 30-40.000 km (specialmente su un motore GPL), conviene sostituirle;
- *bobina di accensione del cilindro 1*; molto comune. Inizialmente il difetto si manifesta sotto sforzo (salita, piena accelerazione), poi peggiora fino a comparire al minimo. Un test

semplice è scambiare la bobina del cilindro 1 con quella del cilindro 2 (se sono singole) e vedere se l'errore diventa P0302;

- *cablaggio o connettore della bobina*; un falso contatto può causare misfire intermittenti;
- *iniettore benzina del cilindro 1*; è meno probabile, perché il problema compare anche a GPL. Tuttavia un iniettore bloccato aperto potrebbe influire anche durante il funzionamento a GPL in alcune condizioni;
- *compressione bassa del cilindro 1*; valvola bruciata o usurata (nei motori GPL non è raro), fasce elastiche consumate o guarnizione della testata;
- *perdita d'aria vicino al cilindro 1*; collettore di aspirazione o guarnizione.

Dal modo in cui è peggiorato...
La sequenza che descrivi è abbastanza tipica:

- *fase iniziale*; misfire solo sotto forte carico;
- *fase intermedia*; poi anche in accelerazione normale;
- *fase attuale*; infine anche al minimo.

Questo andamento è molto compatibile con una bobina che sta cedendo oppure una candela ormai esausta.
Cosa farei io, in ordine

- *diagnosi elettronica*; leggere anche i dati "freeze frame" e verificare se sono presenti altri codici;
- *controllo candela*; controllare la candela del cilindro 1;
- *scambio bobine*; scambiare la bobina del cilindro 1 con quella del cilindro 2;
- *verifica errore bobina*; se l'errore diventa P0302, hai trovato il colpevole;
- *prova di compressione*; se l'errore resta P0301, fare una prova di compressione;
- *controlli secondari*; se la compressione è buona, controllare l'iniettore e il cablaggio.

Alcune domande che aiuterebbero a restringere il campo

- *motorizzazione*; che motore monta la Sportage? 2.0 16V da 141 CV oppure un altro?
- *chilometraggio*; quanti chilometri ha?
- *manutenzione*; candele e bobine sono mai state sostituite?
- *comportamento spia*; la spia motore lampeggia quando va a tre cilindri oppure rimane fissa?
- *test scollegamento*; quando togli il connettore della bobina del cilindro 1 a motore acceso, il regime cambia oppure resta praticamente uguale?

Con queste informazioni si può capire se è più probabile un problema di accensione oppure un problema meccanico (ad esempio una valvola del cilindro 1).

Il modello di Intelligenza Artificiale mostra un ragionamento coerente e individua le cause che possono celarsi dietro al guasto, disponendole in una lista in ordine di priorità.

Utente

Ho appena acquistato e installato delle candele nuove e scambiato le due bobine ma l'errore rimane P0301. Non ho invertito i cavi perché il cilindro 1 ha il cavo più lungo di tutti

Di seguito il risultato:

ChatGPT

Questa informazione è molto utile. Se:

- *candele*; hai montato candele nuove;
- *bobine*; hai scambiato fisicamente le bobine;
- *codice errore*; l'errore rimane sempre P0301;
- *alimentazione*; il difetto è presente sia a benzina sia a GPL;

allora la probabilità che sia la bobina è molto bassa.
Rimangono principalmente queste possibilità
1. Problema nel cablaggio della bobina del cilindro 1
Anche se la bobina è stata scambiata, il connettore e i fili che la alimentano sono rimasti sul cilindro 1.
Controlla:

- *stato connettore*; connettore ossidato;
- *integrità cablaggio*; fili spezzati vicino al connettore;
- *alimentazione*; tensione di alimentazione alla bobina;
- *segnale centralina*; comando dalla centralina motore.

Se hai un multimetro, puoi già fare alcune verifiche.
2. Compressione bassa del cilindro 1
Su un motore alimentato anche a GPL è una delle ipotesi più importanti.
Un cilindro con una valvola di scarico che non chiude bene può dare:

- *comportamento al minimo*; misfire al minimo;
- *comportamento sotto carico*; peggioramento sotto sforzo;
- *persistenza errore*; P0301 persistente.

Serve una prova di compressione oppure, ancora meglio, un leak-down test.
3. Iniettore benzina del cilindro 1
È meno probabile perché il problema c'è anche a GPL.
Tuttavia, se l'iniettore benzina perde e rimane leggermente aperto, può alterare la miscela anche quando il motore funziona a GPL.
4. Perdita di aspirazione sul cilindro 1
Una guarnizione del collettore d'aspirazione danneggiata vicino al cilindro 1 può causare un misfire soprattutto al minimo.
Vorrei capire una cosa importante
Hai scritto: "Non ho invertito i cavi perché il cilindro 1 ha il cavo più lungo."
Questa frase mi fa pensare che il tuo motore non abbia bobine singole montate direttamente sulle candele, ma una bobina doppia con cavi candela.
Puoi dirmi:

- *tipo di motore*; il motore è il 2.0 benzina 16V?
- *configurazione accensione*; hai una sola bobina con quattro cavi, oppure due bobine, oppure quattro bobine singole?

Un test molto semplice
Con il motore al minimo:

- *test cilindro 1*; scollega il connettore dell'iniettore benzina del cilindro 1 e osserva se il motore peggiora;

ChatGPT

- *test cilindro 2*; poi fai la stessa prova sul cilindro 2.

Se scollegando il cilindro 1 non cambia praticamente nulla, mentre scollegando il 2 il motore peggiora molto, significa che il cilindro 1 sta già contribuendo poco o nulla. A quel punto bisogna capire se il motivo è elettrico, di alimentazione o meccanico.
La mia impressione
Poiché il difetto è iniziato solo sotto forte carico ed è peggiorato progressivamente fino a essere presente anche al minimo, e hai già escluso candele e bobine, le due ipotesi che considero più probabili sono:

- *ipotesi meccanica*; compressione bassa sul cilindro 1 (valvola di scarico usurata o bruciata);
- *ipotesi elettrica*; problema elettrico nel cablaggio della bobina o nel comando della centralina.

Una domanda: quanti chilometri ha la Sportage? Se supera i 180.000-200.000 km e ha percorso molti chilometri a GPL, il controllo della compressione diventa una priorità.

L'utente risponde alle domande a lui poste dal modello e inserisce informazioni aggiuntive.

Utente

Ho 2 bobine in totale e il motore è 2.0 benzina con 180.000km di percorrenza. Aggiungo un'informazione: se percorro la strada in pianura e tengo il motore da 2500giri/min in su, il motore funziona bene

Di seguito il risultato:

ChatGPT

Questa informazione cambia parecchio la diagnosi. Con 2.0 benzina, 180.000 km, GPL, due bobine e cavi candela, il fatto che sopra i 2500 giri/min in pianura funzioni regolarmente è molto significativo.
In particolare, il fatto che il P0301 rimanga sul cilindro 1 anche dopo aver scambiato le due bobine rende la bobina molto meno probabile. Il fatto che il difetto sia molto più evidente a basso regime e sotto carico fa invece pensare soprattutto a un problema che si manifesta quando il cilindro ha meno margine per mantenere la combustione.
La prova che farei prima di tutto: i cavi candela
Dato che hai due bobine ma cavi candela separati, i cavi sono ancora completamente da verificare.
E soprattutto non è necessario invertire il cavo del cilindro 1 con quello di un altro cilindro. Hai ragione: essendo di lunghezze diverse, non avrebbe senso usarli permanentemente invertiti.
Per il tuo caso, visto che il difetto è sul cilindro 1, se il cavo del cilindro 1 avesse una resistenza molto maggiore degli altri, sarebbe un indizio fortissimo.
Però c'è un'altra cosa che mi preoccupa
180.000 km + molti km a GPL + misfire che inizialmente compariva solo sotto forte carico e che ora compare anche al minimo è una combinazione che rende importante escludere una perdita di compressione del cilindro 1, in particolare una possibile valvola che non chiude più perfettamente.
Non significa che sia sicuramente la valvola. Anzi, prima controllerei l'accensione perché è molto più semplice.
Io procederei così:

- *misurazione resistenza cavo 1*; esclude o conferma un problema al cavo candela;
- *controllo connettori e cablaggio bobine*; esclude un eventuale problema di natura elettrica;

- *controllo candele dopo funzionamento*; può mostrare evidenti differenze di combustione tra i vari cilindri;
- *compressione cilindri*; fondamentale per escludere in modo definitivo un problema meccanico;
- *leak-down test*; se la compressione risulta bassa, individua esattamente il punto in cui avviene la perdita.

Un particolare molto interessante
Il tuo comportamento:

- *regime elevato (2500+ rpm)*; il motore gira in modo regolare;
- *bassi giri e carico (salita/carico)*; compare l'errore P0301 e il misfire;

non significa necessariamente che il cilindro 1 "funzioni bene" sopra 2500 giri. Potrebbe semplicemente essere che a quel regime la combustione sia più stabile e il misfire diventi molto meno frequente.
Inoltre, quando sei in salita a basso regime e chiedi molta coppia, la pressione nel cilindro durante la fase di compressione aumenta, rendendo più difficile accendere la miscela se l'impianto di accensione ha un difetto. Questo è proprio il motivo per cui un cavo candela o una candela possono funzionare apparentemente bene a vuoto e fallire sotto carico.

L'utente decide di procedere in autonomia misurando le resistenze dei cavi delle candele e presenta il seguente prompt:

Utente

Consultando il manuale di riparazione, nella sezione relativa alla guida per misurazione della resistenza elettrica dei cavi di accensione delle candele, è stata rilevata questa informazione: "la resistenza dei cavi non dovrebbe essere più alta di 5.6 kΩ per metro di cavo con scarto di +-20%. Se superiore a 10 kΩ per metro bisogna sostituire i cavi". Ebbene, sul cavo lungo 68cm del cilindro numero 1, ho una resistenza totale pari 9.6 kΩ.

Di seguito il risultato:

ChatGPT

Sì: queste misurazioni sono molto significative e, alla luce dei sintomi che hai descritto, cambiano decisamente la diagnosi.
La cosa importante è che il manuale specifica la resistenza per metro, quindi non bisogna confrontare direttamente i 9.7 kΩ del cavo 1 con i 5.6 kΩ indicati dal manuale: bisogna normalizzare in base alla lunghezza.
Calcoliamo la resistenza per metro

- *cavo 1 (C1)*; resistenza 9.7kΩ, lunghezza 0.68 m, resistenza/m 14.2kΩ/m (limite manuale >10 → sostituire);
- *cavo 2 (C2)*; resistenza 7.1 kΩ, lunghezza 0.56 m, resistenza/m 12.68 kΩ/m (limite manuale >10 → sostituire);
- *cavo 3 (C3)*; resistenza 6.8 kΩ, lunghezza 0.47 m, resistenza/m 14.47 kΩ/m (limite manuale >10 → sostituire);
- *cavo 4 (C4)*; resistenza 5 kΩ, lunghezza 0.39 m, resistenza/m 12.8 kΩ/m (limite manuale >10 → sostituire).

Quindi tutti e quattro i cavi superano nettamente i 10 kΩ/m, cioè il limite oltre il quale il manuale prescrive la sostituzione.

E c'è un dettaglio particolarmente interessante: il cavo del cilindro 1 è effettivamente molto resistivo, con 14.26 kΩ/m.
Questo spiega molto bene i tuoi sintomi
Il tuo quadro era:

- *codice errore*; P0301 → sempre cilindro 1;
- *sostituzione candele*; candele nuove → problema invariato;
- *inversione bobine*; bobine scambiate → rimane P0301;
- *carburante*; benzina e GPL → problema presente con entrambi;
- *sintomi iniziali*; inizialmente misfire soprattutto a pieno carico/in salita;
- *sintomi attuali*; adesso misfire anche al minimo;
- *regime elevato*; sopra 2500 giri/min in pianura il motore sembra funzionare bene;
- *chilometraggio*; 180.000 km;
- *misurazione C1*; cavo C1 = 14.26 k Ω/m.

Un cavo candela con resistenza eccessiva può ancora riuscire a far scoccare la scintilla in alcune condizioni, ma può avere maggiori difficoltà quando aumenta la richiesta di accensione, ad esempio sotto carico e con elevata pressione nel cilindro.
Questo rende particolarmente plausibile la sequenza che hai osservato: inizialmente il problema solo in salita a pieno acceleratore, poi progressivamente anche in altre condizioni.
Ma c'è un'altra cosa importante
Non è soltanto il C1 a essere fuori specifica.
Sono fuori specifica tutti e quattro i cavi.
Infatti:

- *cilindro 1 (C1)*; 14.26 k Ω/m;
- *cilindro 2 (C2)*; 12.68 k Ω/m;
- *cilindro 3 (C3)*; 14.47 k Ω/m;
- *cilindro 4 (C4)*; 12.82 k Ω/m.

Sono tutti oltre il limite di 10 kΩ/m indicato dal manuale.
Quindi io, prima di fare una prova di compressione o iniziare a smontare il motore, sostituirei tutti e quattro i cavi candela.
E non mi preoccuperei del fatto che il C1 sia il più lungo: la specifica del manuale è proprio espressa in kΩ per metro, quindi tiene conto della lunghezza.
Una cosa interessante
Il fatto che il C3 abbia 14.47 kΩ/m, quindi leggermente peggiore del C1, non contraddice il P0301. La resistenza in ohm non è necessariamente l'unico parametro che determina la capacità del cavo di far scoccare correttamente la scintilla. Inoltre il difetto può dipendere da una combinazione di resistenza, isolamento e comportamento del cavo sotto alta tensione.
In ogni caso, secondo il criterio del tuo manuale, tutti e quattro sono da sostituire.
Dopo la sostituzione, cancellerei il P0301, farei una prova su strada nelle stesse condizioni in cui si manifestava il problema e controllerei se il codice ritorna.
Se vuoi, possiamo anche confrontare questi valori con quelli dei cavi nuovi dopo la sostituzione e verificare matematicamente se sono nella norma.

Aggiungendo le informazioni richieste, il modello effettua un'inversione dello scenario di diagnosi e guida l'utente nella direzione del guasto ai cavi delle candele suggerendo una misurazione della loro resistenza elettrica, avvalendosi di un multimetro. L'utente, dopo aver consultato ulteriormente il manuale ed effettuato la misurazione della resistenza elettrica, inserisce i dati. Il modello genera una risposta e l'utente acquistando, misurando e sostituendo

i cavi dell'automobile, giunge alla soluzione del problema.

Utente

Ho sostituito i cavi delle candele e ho effettuato la misurazione dei cavi nuovi:

- c1 : 3.8 k Ω 68cm
- c2 : 3.1 k Ω 56cm
- c3 : 2.6 k Ω 47cm
- c4 : 2 k Ω 39cm

Dopo la sostituzione dei cavi il problema del motore è stato risolto.

Di seguito il risultato:

ChatGPT

Perfetto: la diagnosi era quindi corretta. I cavi candela erano effettivamente la causa del P0301 e del funzionamento irregolare del motore.
Confrontando i vecchi e i nuovi cavi:

- *cavo 1 (C1)*; vecchio 9.7 k Ω / 68 cm (14.3 kΩ/m), nuovo 3.8 kΩ / 68 cm (5.6 kΩ/m);
- *cavo 2 (C2)*; vecchio 7.1 k Ω / 56 cm (12.7 kΩ/m), nuovo 3.1 kΩ / 56 cm (5.5 kΩ/m);
- *cavo 3 (C3)*; vecchio 6.8 k Ω / 47 cm (14.5 kΩ/m), nuovo 2.6 kΩ / 47 cm (5.5 kΩ/m);
- *cavo 4 (C4)*; vecchio 5.0 k Ω / 39 cm (12.8 kΩ/m), nuovo 2.0 kΩ / 39 cm (5.1 kΩ/m).

I nuovi cavi sono quindi tutti intorno a 5.4 kΩ/m, praticamente coincidenti con il valore di riferimento di 5.6 kΩ del manuale.
La cosa interessante è che il problema sembrava "meccanico"
Il comportamento che avevi descritto era molto indicativo:

- *comportamento iniziale*; a pieno carico in salita → misfire;
- *evoluzione*; poi anche al minimo;
- *funzionamento*; funzionamento alternato 4/3 cilindri;
- *codice errore*; P0301;
- *carburante*; sia benzina che GPL;
- *regime elevato*; sopra 2500 rpm in pianura → apparentemente regolare.

Era facile pensare a una valvola del cilindro 1 o a un problema di compressione, soprattutto considerando i 180.000 km e l'utilizzo GPL.
Invece il problema era nell'accensione.
Il motivo per cui i cavi potevano causare un difetto apparentemente "strano" è che un sistema di accensione può funzionare in alcune condizioni e fallire in altre. Sotto carico la tensione necessaria per far scoccare la scintilla aumenta; un cavo deteriorato può quindi provocare mancate accensioni proprio quando il motore ha maggiore richiesta.
E il fatto che il P0301 non si sia spostato quando hai scambiato le bobine era un indizio importante: il problema rimaneva sul lato del cilindro 1, quindi bisognava continuare a valle della bobina, cioè verso cavo e candela.
Diagnosi finale

- *causa*; cavi di accensione fortemente degradati, tutti oltre il limite di 10 kΩ/m previsto dal manuale;
- *intervento*; sostituzione dei 4 cavi candela;

- *risultato*; motore nuovamente regolare e P0301 risolto.

Direi quindi che puoi considerare la riparazione riuscita.
E soprattutto hai fatto bene a misurare i cavi prima di sostituirli: hai una conferma strumentale molto più forte del semplice "ho cambiato il pezzo e ora funziona".

3.4 Analisi comparativa dei risultati

In seguito alle risposte dei due modelli interrogati, si desidera effettuare una valutazione più metodologica elencando come si sono distinti i due modelli di Intelligenza Artificiale nelle varie situazioni di diagnostica.

Nella valutazione qualitativa verranno elencati i criteri di valutazione che abbiamo definito in precedenza effettuando un commento per ogni criterio.

Nella Tabella 3.1 è riportata la valutazione qualitativa di come si sono destreggiati i due modelli nel case study in analisi, mentre nella Tabella 3.2 è riportata la valutazione quantitativa.

	Gemini	**ChatGPT**
Pertinenza al contesto	Presenta una buona pertinenza al contesto	Presenta un'ottima pertinenza al contesto
Presenza di errori	Presenti nei calcoli delle resistenze elettriche	Del tutto assente
Deriva conversazionale	Presente dopo un'esclusione di causa errata	Contenuta
Soluzione al problema	Esito positivo	Esito positivo

Tabella 3.1: Valutazione qualitativa dei modelli di Intelligenza Artificiale

	Gemini	**ChatGPT**
Numero di interazioni	5	5
Riformulazioni dei prompt	1	Nessuna
Customer satisfaction score	3/5	4/5
Customer effort score	3/5	2/5

Tabella 3.2: Valutazione quantitativa dei modelli di Intelligenza Artificiale

3.5 Discussione, limiti e linee guida pratiche

L'esperimento di diagnostica comparativa con i due modelli di Intelligenza Artificiale sul guasto automobilistico reale evidenzia, in entrambi i sistemi, elevate capacità analitiche. Risulta, tuttavia, evidente la presenza di limiti nella stima delle probabilità, giustificati sia dall'impossibilità di analizzare direttamente il veicolo, sia dalla necessità di basarsi esclusivamente sulle osservazioni personali dell'utente.

Entrambi i modelli hanno dimostrato una comprensione del contesto operativo:

- *Interpretazione dei sintomi*; i modelli sono stati in grado di escludere gli iniettori dalle possibili cause, comprendendo che il problema si manifesta sia a GPL che a benzina;

- *gestione dei dati tecnici*; i modelli sono riusciti a interpretare correttamente i dati tecnici relativi alla resistenza elettrica misurata nei quattro cavi delle candele.

La principale differenza è emersa nella gestione delle priorità diagnostiche:

- *Gemini*: in un primo momento, ha mostrato più interesse per il problema elettrico, per poi orientarsi verso il guasto meccanico più grave (il gioco delle valvole) che, come verificato in seguito, non è stato la causa del problema.
- *ChatGPT*: ha mantenuto un'interazione più intuitiva e progressiva; inizialmente ha valutato le due cause principali a pari merito, per poi sbilanciarsi, in una fase successiva, verso quella effettivamente più probabile.

3.5.1 Limiti dell'Intelligenza Artificiale nella diagnostica meccanica

I dataset utilizzati per l'addestramento dei sistemi di Intelligenza Artificiale sono di vari formati: testo, immagini, video e audio. L'Intelligenza Artificiale non è limitata soltanto dalla quantità di parametri e dalla loro ottimizzazione, ma anche dal flusso di informazioni che può rilevare. Al momento, questi sistemi si stanno scontrando con il cosiddetto *Data Wall* che rappresenta il vero limite. La quasi totalità dei sistemi di Intelligenza Artificiale ha ormai acquisito tutti i manoscritti umani disponibili. Ciò crea un *bottleneck* in cui questi sistemi non hanno più fonti per l'addestramento, nonostante la loro grande capacità di apprendimento.

Risulta, quindi, necessario che, per l'addestramento, si utilizzino dati in formato video. Se un modello di Intelligenza Artificiale fosse in grado di visionare un video, apprendendo non soltanto gli aspetti basilari, ma analizzando anche il contesto al contorno, potrebbe incrementare l'estrazione di dati utili per la formulazione di un dataset più approfondito.

I principali limiti riscontrati durante il confronto dei modelli linguistici con il mondo fisico sono i seguenti:

- *Sovrastima delle cause*; i modelli hanno preso molto in considerazione i guasti che caratterizzano le automobili a GPL con un alto chilometraggio, considerandoli come causa più probabile.
- *Deriva conversazionale nel branch secondario*; come evidenziato durante il test, Gemini ha suggerito di procedere subito con il test più invasivo, scartando la verifica più semplice.
- *Dipendenza dai dati strumentali*; i modelli non possiedono percezioni sensoriali umane e dipendono esclusivamente dai dati comunicati in modo esplicito. Se non fosse stato per l'utente, che ha deciso di sua iniziativa di effettuare un test a costo zero, l'indagine del guasto si sarebbe probabilmente spostata su una causa secondaria.

3.5.2 Linee guida pratiche

Durante la sperimentazione con i due modelli di Intelligenza Artificiale è stato necessario lo sviluppo di alcune linee guida per avere delle risposte con un tasso di allucinazioni minore e, quindi, una qualità generale maggiore. Vediamo le linee guida individuate che hanno portato al risultato atteso:

- *Principio di economicità*; il modello, da quello che è stato rilevato, non sempre riesce a comprendere i vincoli di fattibilità economica; un'ulteriore verifica di un componente sospetto del malfunzionamento a costo zero è preferibile a un intervento oneroso.

- *Dati tecnici quantitativi*; una descrizione troppo generica del problema può disorientare un modello di Intelligenza Artificiale; è quindi necessario introdurre dati concreti per aiutare il modello nell'elaborazione del contenuto.
- *Controllo umano costante*; come abbiamo visto, l'attore utente deve mantenere un ruolo primario nelle interazioni; questo permette una compensazione degli aspetti umani che mancano all'Intelligenza Artificiale.
- *Branching*; se si vogliono esplorare soluzioni alternative, è raccomandato effettuare una diramazione in una chat secondaria; questo permette di valutare una potenziale soluzione al problema, senza perdere il contesto del branch primario.

3.6 Conclusione

Questo intervento si distingue per un processo di diagnostica impeccabile; partendo da una sintomatologia meccanica complessa, ma precisa, si sono progressivamente escluse le cause più comuni, fino a isolare l'origine del malfunzionamento dell'automobile. I risultati ottenuti dimostrano chiaramente che i modelli presi in analisi costituiscono un valido strumento analitico e che la metodologia applicata è perfettamente replicabile fornendo margini di miglioramento futuri.

CAPITOLO 4

Confronto degli LLM nella gestione delle immagini

Nel capitolo precedente abbiamo visto come due modelli di Intelligenza Artificiale, mediante prompt testuali, possono essere utilizzati per informarsi, diagnosticare e riparare un'automobile. In questo capitolo cambiamo totalmente approccio e utilizziamo i Large Language Model intrinsecamente multimodali. L'obiettivo è la prototipazione e la realizzazione di interfacce utente per un'applicazione che può essere impiegata in molteplici contesti.

4.1 Introduzione e definizione del problema

In questa sezione procederemo, in primo luogo, con la definizione e il significato tecnico di un artefatto visivo creato dall'Intelligenza Artificiale, per poi proseguire con la definizione del problema e il relativo prompt iniziale, da sottoporre al modello.

4.1.1 Artefatti visivi delle immagini

Un artefatto visivo è una difformità tra la realtà fisica e la struttura di un'immagine, creata mediante l'utilizzo dell'Intelligenza Artificiale Generativa. Esso possiede la particolarità di essere non intenzionale ed è caratterizzato da pattern ripetitivi in contesti generativi di particolare complessità strutturale e semantica, come, ad esempio, la rappresentazione della struttura di una mano, oppure la generazione di testo. Gli artefatti visivi possono essere considerati, quindi, come degli errori dell'immagine, non vengono generati casualmente, ma sono direttamente correlati con le modalità con cui un sistema di Intelligenza Artificiale Generativa sviluppa un file immagine.

Di seguito elenchiamo le cause più probabili in cui si manifestano questi errori visivi:

- *Sovraccarico cognitivo*: nel momento in cui, un prompt testuale presenta un accumulo elevato di vincoli, un sistema di Intelligenza Artificiale perde velocemente la capacità di sintetizzare correttamente l'ambiente richiesto dal prompt, presentando incoerenze strutturali come l'aggiunta, ad esempio, di sei dita sulle mani.
- *Ragionamento spaziale mancante*: i modelli di Intelligenza Artificiale incontrano serie difficoltà nel comprendere aspetti come la gravità e la tridimensionalità del mondo fisico, attuando dei meccanismi di stima per pixel, che compongono l'immagine generata.
- *Spazio latente*: durante la generazione di un'immagine, per la complessità computazionale, essa non viene generata pixel per pixel, ma si sviluppa tramite un processo con il *variational autoencoder*; questo, però, genera elementi estranei al contesto del prompt.

Una classificazione strutturata delle anomalie che possono presentarsi durante la generazione di un'immagine è utile all'analisi dell'errore che ognuna di esse può presentare.

Le anomalie sono presenti, soprattutto, nella generazione di immagini che raffigurano umani o animali, in quanto essi presentano molti gradi di libertà anatomici.

L'Intelligenza Artificiale, poiché utilizza meccanismi di generazione predittivi, avrà difficoltà nel generare un'immagine fedele alla nostra richiesta, vista l'infinità di combinazioni di posizioni articolari.

Di seguito presenteremo le principali strutture anatomiche complesse, in cui l'AI tende a generare errori:

- *Simmetria del volto*: l'essere umano presenta un volto con molte espressioni facciali, un modello di Intelligenza Artificiale presenta serie difficoltà nel momento in cui gli si chiede di generare espressioni facciali veritiere, dinamiche e credibili.

- *Polidattilia*: a seguito della richiesta di raffigurazione delle mani, un modello può generare un numero eccessivo di dit; ciò è dovuto principalmente all'addestramento. Come descritto nel Capitolo 3, se un modello di Intelligenza Artificiale potesse estrarre e imparare la rappresentazione semantica della struttura delle immagini di addestramento, non avrebbe questo problema.

- *Asimmetria oculare*: sebbene i modelli di Intelligenza Artificiale presentano un'ottima produzione di strutture, come la pelle umana, hanno serie difficoltà nella generazione di uno sguardo nella stessa direzione tra due occhi; per questo le immagini generate presentano spesso assi visivi dei due occhi divergenti.

Nei software in cui è presente, il ray tracing, ovvvero la determinazione della rifrazione o delle ombreggiature, segue regole coerenti con le leggi fisiche; questo permette di determinare, in un modello tridimensionale, un'ombreggiatura veritiera con la realtà.

L'Intelligenza Artificiale Generativa non possiede il modello tridimensionale dell'ambiente che verrà generato. Di seguito, elencheremo i fenomeni rilevati e causati da questa mancanza:

- *Fisica della luce*: il modello genera riflessi e ombre incoerenti; ciò è dovuto dalla generazione alla immagini in maniera statistica. La creazione delle ombre e dei riflessi si riduce alla semplice applicazione dei pattern appresi, senza l'applicazione delle leggi fisiche.

- *Prospettiva spaziale paradossale*: gli oggetti presentano forme e dimensioni incoerenti, questo porta ad avere un'immagine che, a primo impatto, sembra reale, tuttavia, nel momento in cui ci si sofferma sui singoli particolari, si nota immediatamente la difformità con l'oggetto reale.

- *Rimescolamento semantico*: questo fenomeno è particolarmente noto nella generazione delle immagini e si presenta nel momento in cui due corpi estranei vengono fusi in un unico oggetto sfumando i confini. Esso può verificarsi, ad esempio, con la fusione dei gioielli, come le collane, con la pelle del collo.

Un altro fenomeno che si presenta con cadenza regolare è la generazione delle immagini con pseudo-testo, come illustrato nella Figura 4.1.

I modelli di Intelligenza Artificiale, durante l'addestramento, non apprendono la grammatica, ma soltanto le posizioni del testo all'interno di un ambiente.

Durante la costruzione dell'immagine, un modello, tramite operazioni statistiche, ricostruisce forme pseudo testuali. Per ovviare a questo problema, recentemente è stata introdotta



Figura 4.1: Immagine contenente un esempio di pseudo testo

una tecnologia basata su text encoder avanzati, che sono in grado di leggere e comprendere i caratteri di testo scritto e generare in modo nitido le scritte.

4.1.2 Obiettivo della sperimentazione

Il presente case study valuta le capacità multimodali dei due principali Large Language Model: ChatGPT e Gemini, nell'elaborazione di un prompt testuale in un'immagine. La prova pratica riguarda la realizzazione, in formato immagine, di un'interfaccia utente per la console centrale di controllo delle attrezzature all'interno di un'ambulanza.

Il mezzo di soccorso è composto da due ambienti di lavoro, ovvero:

- *vano autista*; è l'ambiente in cui è presente l'autista soccorritore che ha accesso al terminale che gestisce, a sua volta, le attrezzature del vano, come i lampeggianti e sirene;
- *vano sanitario*; è l'ambiente in cui, solitamente, è presente il soccorritore secondario e ha accesso al secondo terminale di gestione delle attrezzature presenti all'interno al vano.

Possiamo suddividere le attrezzature presenti nei due ambienti, che sono accessibili dall'interfaccia utente sul dispositivo, in due liste distinte.

Le attrezzature direttamente accessibili dalla console centrale nel vano autista sono:

- *lampeggianti e sirene*;
- *climatizzazione*;
- *comunicazione con la centrale operativa*;
- *illuminazione del vano autista e del vano sanitario*.

Le attrezzature direttamente accessibili dalla console centrale del vano sanitario sono le seguenti:

- *illuminazione vano sanitario*;
- *climatizzazione e termoregolazione dei farmaci*;
- *aspiratore integrato*;
- *gestione elettrica e inverter*;
- *impianto di ventilazione e sanificazione*.

L'obiettivo primario è la continuazione di un'interfaccia utente che sia graficamente compatibile con le attrezzature da controllare.

Per effettuare un confronto oggettivo tra i modelli multimodali di ChatGPT e Gemini, che sono stati presi in analisi, è stata definita una metodologia di testing empirica tramite prompt e valutazione dell'immagine generata.

Per garantire che i modelli effettueranno l'elaborazione del prompt senza condizionamenti dovuti alle prove passate, verranno eliminati tutti i dati relativi alle sessioni di prova precedenti e verrà generata una nuova sessione.

La valutazione delle risposte generate dagli LLM si articola su quattro metriche principali:

- *aderenza al prompt*: misura la capacità del modello di individuare, all'interno del testo, tutti i vincoli per la progettazione dell'interfaccia utente;
- *qualità del dettaglio e assenza di allucinazioni visive*: misura se il modello di Intelligenza Artificiale è stato in grado di generare immagini coerenti con la realtà;
- *coerenza spaziale e composizione*: valuta la correttezza delle disposizioni degli elementi nello spazio e il rispetto delle proporzioni dei pulsanti;
- *creatività contestuale*: misura la capacità del modello di comprendere concetti complessi e di usare la propria creatività per presentare la propria soluzione.

Ad ogni metrica di valutazione verrà attribuito un punteggio numerico da 1 a 5; questo permetterà di prendere visione su come ogni modello di Intelligenza Artificiale Generativa si destreggia con l'esecuzione dei prompt forniti dall'utente.

4.1.3 Progettazione del prompt iniziale

A ciascun modello viene comunicato lo stesso prompt testuale iniziale, questo ci garantisce che entrambi abbiamo lo stesso input tuttavia, visto che ogni rete neurale dei modelli considerati è unica e inaccessibile per una visura, molto probabilmente verranno generate delle soluzioni, anch'esse, uniche.

Di seguito riportiamo il prompt iniziale che verrà utilizzato:

Utente

Crea un'interfaccia utente di un'app per tablet che sostituisca i tasti hardware, per la gestione delle varie attrezzature presenti all'interno di un'ambulanza.
Deve essere costruita in modo che sia di facile utilizzo durante le operazioni e il movimento dell'ambulanza.

Il prompt iniziale, relativo al nostro case study, è generico nella parte iniziale, in cui si lascia al sistema di Intelligenza Artificiale la scelta di come progettare l'interfaccia utente. Esso è, invece, specifico nella sezione in cui si chiede di prendere in considerazione le modalità operative in cui verrà utilizzata.

I prompt successivi al primo saranno specifici per ogni modello, a seconda del risultato del prompt iniziale.

4.2 Esecuzione del test

In questa sezione procederemo con il processo di sperimentazione dei due modelli di Intelligenza Artificiale Generativa presi in analisi: Gemini e ChatGPT.

Per ciascuno scenario, verranno presentati i prompt, seguiti dalla risposta dei modelli nei termini di immagine generata.

4.2.1 Scenario di test con Gemini

Illustriamo di seguito il risultato generato interrogando il sistema Gemini con il prompt iniziale:

Gemini



Utente

Ora taglia lo sfondo e i bordi del tablet dall'immagine lasciando solamente l'interfaccia grafica dell'app

Di seguito l'immagine generata:

Gemini



Utente

Ridimensiona l'interfaccia utente in modo che occupi tutto lo spazio utile appena creato tagliando lo sfondo.

Di seguito l'immagine generata:

Gemini



Utente

Modifica i seguenti riquadri:

- *Luci d'emergenza*; inserisci un pulsante per le sirene e 3 modalità per i lampeggianti: spenti, acceso fisso e acceso lampeggiante;
- *Comunicazione*; rimuovi pulsante della mappa GPS e inserisci un pulsante per chiamare la centrale operativa;
- *Centro dati paziente*; rimuovi il nome e inserisci una mini mappa di navigazione.

Di seguito l'immagine generata:

Gemini



4.2.2 Scenario di test con ChatGPT

Illustriamo di seguito il risultato generato interrogando il sistema ChatGPT con il prompt iniziale:

ChatGPT



Utente

Ora taglia lo sfondo e i bordi del tablet dall'immagine lasciando solamente l'interfaccia grafica dell'app

Di seguito l'immagine generata:

ChatGPT



Utente

Ora al posto del riquadro dell'ossigeno metti dei pulsanti per accendere sirene e lampeggianti sopra l'ambulanza; i lampeggianti devono avere 3 modalità: spento, acceso fisso, acceso lampeggiante,

Di seguito l'immagine generata:

ChatGPT



Utente

Rimuovi il menù laterale a sinistra dell'interfaccia e ridimensiona i riquadri degli strumenti in modo da occupare anche lo spazio utile creato

Di seguito l'immagine generata:

ChatGPT



Utente

Nel riquadro della climatizzazione, rimuovi le modalità di funzionamento e metti due pulsanti per selezionare la climatizzazione del vano sanitario e del vano anteriore.

Di seguito l'immagine generata:

ChatGPT



4.3 Analisi comparativa

In questa sezione è presentata l'analisi comparativa tra i due modelli di Intelligenza Artificiale nella generazione di immagini da un prompt testuale.

Utilizzando le metriche con i relativi punteggi, in Tabella 4.1 vengono presentati i risultati della sperimentazione con Gemini e ChatGPT.

	Gemini	**ChatGPT**
Aderenza al prompt	5	5
Qualità del dettaglio e assenza di allucinazioni visive	3	3
Coerenza spaziale e composizione	3	3
Creatività contestuale	3	4

Tabella 4.1: Valutazione quantitativa dei modelli di Intelligenza Artificiale

Considerando la valutazione, è interessante notare che entrambi i modelli hanno presentato risultati simili tra loro. Durante la progettazione delle immagini entrambi generavano errori molto simili, come, ad esempio, la presenza di sfondo indesiderato nell'interfaccia utente.

4.3.1 Confronto delle prestazioni e accuratezza

Entrambe le interfacce del sistema di controllo, tramite console centrale, mostrano un'elevata accuratezza nella rappresentazione dei moduli di strumenti dell'ambulanza.

Nell'interfaccia finale generata da Gemini, vi è uno stile colorato e diviso per blocchi.

Ogni sezione è ben delimitata; questo favorisce la rapida individuazione, durante l'operatività del mezzo, permettendo di selezionare le attrezzature necessarie. Le informazioni, come la temperatura di climatizzazione e la mappa di navigazione, sono raffigurate con un layout ben organizzato e in modo intuitivo.

L'interfaccia finale presentata da ChatGPT, invece, possiede un design più moderno e un'estetica scura in stile dark-mode. Le informazioni presentate sull'immagine sono esaustive e ben dettagliate, permettendo, anche qui, l'uso facilitato durante l'operatività dell'ambulanza.

In termini di accuratezza, sia Gemini che ChatGPT hanno presentato due soluzioni che illustrano in modo fedele gli elementi e gli strumenti che ci si aspetterebbe di trovare nelle ambulanze di soccorso, come le sirene, i dati del paziente, la comunicazione radio e la gestione dei sistemi a bordo. In entrambi i casi, le interfacce risultano realistiche e vicine alla realtà.

In sintesi, anche se le due interfacce presentate risultano completamente diverse tra loro, adottando due stili completamente differenti, presentano un layout ben strutturato con un'accurata raccolta di informazioni e tasti di controllo per un'interfaccia di questo genere.

4.3.2 Limiti riscontrati ed errori ricorrenti

Analizzando le interazioni tra utente e modelli di Intelligenza Artificiale, possiamo individuare alcuni limiti ed errori grafici che sono presenti con cadenza frequente.

Gemini presenta spesso errori grafici e problematiche di comprensione con l'utente. Vediamo le problematiche più comuni:

- *Sfondo non eliminabile*: il modello ha necessitato di molteplici prove con prompt differenti, in quanto non riusciva ad eliminare lo sfondo presente ma raffigurava in continuazione lo sfondo grigio.
- *Incongruenze di significato*: la presenza del pulsante per l'accensione dei fari abbaglianti da tablet è particolarmente controintuitiva.
- *Riduzione della temperatura assente*: il riquadro della climatizzazione manifesta come unica possibilità l'aumento della temperatura.

ChatGPT, d'altronde, rivela anch'esso problematiche simili:

- *Sovraffollamento informativo*: l'intera interfaccia utente presenta una densità di informazioni che rendono la lettura particolarmente difficile durante l'operatività del mezzo.
- *Pseudo testo*: alcune delle immagini generate possedevano testo incomprensibile, questo nonostante ChatGPT implementasse le tecnologie e i filtri che cercano di prevenire tale fenomeno.

4.4 Conclusioni generali

Durante la sperimentazione e l'analisi finale dei risultati, sono state individuate delle linee guida che hanno aiutato ad incrementare la qualità dell'output generato, vediamo in particolare le più importanti:

- *Nuova sessione*: nel momento in cui si ha una generazione errata del primo prompt e una generazione errata del secondo prompt correttivo, è stato utile avviare una nuova sessione.

- *Definizione dei confini*: durante la progettazione del prompt risulta necessario impostare dei confini in modo che il modello di Intelligenza Artificiale non divaghi;
- *Catena di compiti*: per compiti complessi risulta necessario utilizzare la strategia "divide et impera" durante la scrittura dei prompt secondari per far lavorare il modello a piccoli compiti.

In conclusione, nonostante le difficoltà iniziali dei modelli nella generazione dell'interfaccia utente, l'utilizzo di un approccio basato su reset del contesto, delimitazione chiara dei confini d'azione e scomposizione dei problemi complessi, ha portato alla generazione di una soluzione coerente con il problema presentato.

Le prove effettuate dimostrano che il risultato non dipende tanto dalle capacità di elaborazione del modello, quanto dal modo in cui esso viene guidato permettendo di trasformare un'interfaccia iniziale imprecisa in un'interfaccia funzionale e di alta qualità.

CAPITOLO 5

Confronto degli LLM nella gestione del codice

Nel capitolo precedente abbiamo utilizzato l'Intelligenza Artificiale Generativa per la creazione delle interfacce utente per la gestione delle strumentazioni all'interno delle ambulanze. In questo capitolo procederemo con la generazione di codice, in particolare funzioni che lavoreranno come le API per estrarre dati da tre siti web differenti. L'unica differenza rilevante tra le API e le funzioni che saranno generate dai sistemi di Intelligenza Artificiale è il fatto che verranno eseguite in locale.

Alla fine di ogni test faremo una valutazione di completezza tramite requisiti funzionali e non funzionali; questo permetterà di prendere visione su come si sono distinti i vari modelli in questo compito.

5.1 Obiettivo dello studio e metodologia di approccio

In questa sezione procederemo con la spiegazione dell'obiettivo dello studio e la metodologia di analisi; ciò ci permetterà di comprendere l'oggetto che verrà sviluppato. Viste le modalità di analisi, introdurremo le tecnologie dietro ai siti web che analizzeremo.

5.1.1 Obiettivo del confronto

Desideriamo valutare le capacità di due modelli di Intelligenza Artificiale nella generazione di codice, utilizzando Python come linguaggio di programmazione, avendo come obiettivo primario l'automazione delle operazioni su un sito web che non presenta API pubbliche.

Gli strumenti di Intelligenza Artificiale Generativa non sono più semplici chatbot che fungono da assistenti, ma iniziano ad essere profondamente integrati nei flussi di lavoro aziendali, con lo scopo di accelerare lo sviluppo e ridurre i costi. Integrando questa tecnologia nel flusso di produzione del codice per eseguire automazioni possiamo evitare la correzione manuale del codice ogni qual volta che quest'ultimo non funziona perchè il DOM del sito web automatizzato è stato modificato.

5.1.2 Metodologia di analisi

Il case study verrà sviluppato tramite dei prompt che verranno posti al modello per la generazione di codice.

La prima analisi verrà eseguita sul modello Codex di OpenAI; ad esso verrà fornito un prompt contenente le richieste e i vincoli che verranno strutturati appositamente per ogni sito target stabilito.

Le richieste possono essere attribuite al "cosa" vogliamo che il modello sviluppi, mentre i vincoli possono essere attribuiti al "come" vogliamo che il modello sviluppi.

Dopo la generazione dello script in Python, eseguiremo un'analisi dei due modelli messi a confronto tramite la soddisfazione dei requisiti funzionali e non funzionali, che verranno definiti in un secondo momento. Lo script verrà eseguito in locale e ci restituirà in formato JSON i dati richiesti.

5.1.3 Criterio di selezione dei siti target

La selezione dei siti target svolge un ruolo fondamentale per studiare la capacità dei sistemi di Intelligenza Artificiale nella generazione di codice atto ad interagire con i siti web. Per la selezione è necessario scegliere dei siti web con un potenziale di utilizzabilità e la fattibilità di recupero delle informazioni.

La scelta è ricaduta su tre piattaforme web reali di livello enterprise, che sono state rinominate con dei code-name. L'elenco dei code-name attribuiti ad ogni sito si trova di seguito:

- *TargetEcomA*; il sito A è una piattaforma di e-commerce di testing di automazioni.
- *TargetEcomB*; il sito B è una piattaforma di e-commerce nel settore dell'agricoltura.
- *TargetEcomC*; il sito C è una piattaforma di e-commerce retail nel settore dell'abbigliamento.

Per l'automazione verrà richiesta la generazione di codice utilizzando la libreria Playwright, famosa per il ruolo che possiede per questi compiti.

Nella Tabella 5.1 viene riportata un'analisi tecnologica dei siti web che analizzeremo.

ID target	**Tipologia servizio**	**Tecnologia stimata**	**Sfide per Playwright**
TargetEcomA	E-Commerce B2C	Single page application	Gestione della paginazione
TargetEcomB	E-Commerce B2C	Single page application	Gestione della paginazione, attributi id mancanti
TargetEcomC	E-Commerce B2C/B2B	Single page application, lazy loading	Gestione della paginazione, gestione dei Cookie, barriere antibot

Tabella 5.1: Caratteristiche tecnologiche dei siti considerati

5.1.3.1 Descrizione del target A

Il sito web in analisi è una piattaforma creata appositamente per eseguire automazioni web in un luogo sicuro sandbox; ciò permette il test dei propri algoritmi e di prendere visione esattamente di dove e come ci si riesce a spingere nell'automazione o nell'estrazione dei dati.

La scelta è ricaduta su questo sito perché, come primo test di automazione, si vorrebbe provare come l'Intelligenza Artificiale reagisca ad un sito statico nel tempo. Ciò permette di valutare se, con un sito che riceve pochi aggiornamenti nel tempo, l'Intelligenza Artificiale riesce ad individuare correttamente i selettori e la strategia per raggiungere l'obiettivo desiderato.

Il sito web presenta un'architettura prevalentemente ad albero generico, in cui la navigazione ha inizio dalla home page per poi proseguire ai livelli sottostanti.

Nella sezione inferiore della pagina è presente il meccanismo di paginazione tramite una barra di paginazione numerata.

5.1.3.2 Descrizione del target B

Il sito in analisi è un sito di e-commerce B2C (Business-to-Customer) specializzato nei settori di giardinaggio, pet care e arredo della casa. Lo studio su questo sito è rilevante per capire come un software possa interfacciarsi con una piattaforma classica, con forte stagionalità e aggiornamenti di magazzino regolari.

L'architettura del sito presenta una struttura che segue standard moderni per la categoria dell'e-commerce.

Il sito presenta due principali macro-aree con cui ci interessa interagire:

- *Categorie e navigazione ad albero*: i prodotti presenti nella piattaforma seguono un'organizzazione in categorie e sottocategorie. Lo script per l'automazione può seguire direttamente un link alla categoria d'interesse anzichè la selezione della categoria dalla barra di ricerca tramite parole chiave.
- *La scheda prodotto*: presenta una struttura standardizzata, ma ricca di informazioni, come il codice identificativo, il prezzo di listino, la disponibilità nel magazzino e altri elementi. L'obiettivo principale di un software di automazione è il monitoraggio della disponibilità nel magazzino o la stima della quantità di prodotti disponibili per la vendita.

5.1.3.3 Descrizione del target C

Il sito in analisi è un sito di e-commerce ibrido B2C (Business-to-Customer) e B2B (Business-to-Business) nel settore dell'abbigliamento.

Passare dall'automazione di TargetEcomB, dove sono presenti per lo più pezzi unici, all'automazione del TargetEcomC, dove abbiamo un catalogo industriale standardizzato che presenta algoritmi complessi unici al mondo, richiede un approccio totalmente diverso.

La piattaforma opera come un aggregatore dinamico di oggetti pubblicati da ogni utente o azienda e disponibili nella home page come una griglia di oggetti. L'analisi del sito non si concentra su aspetti commerciali, ma solo sul contenuto informativo e tecnologico che esso presenta.

Il sito presenta principalmente due macro-aree con le quali ci interessa interagire; esse sono:

- *Ricerca e filtraggio degli articoli*: è il punto di partenza per individuare i prodotti che interessano all'analisi, dato che il sito web permette un filtraggio qualitativo per marca o prezzo. Per un software di automazione questo significa dover gestire click ed inserimento dei parametri necessari al filtraggio.
- *La scheda del prodotto*: il software deve riuscire ad estrarre i dati dei prodotti dalle schede, senza andare in errore.

A livello tecnico, il sito web non è statico, ma una piattaforma web dinamica, che all'apertura della pagina procede con il caricamento degli articoli al volo, tramite JavaScript. Ciò permette al sito di gestire lo scrolling infinito, e quindi di caricare, al momento giusto, gli articoli da mostrare all'utente. Questo comportamento limita notevolmente l'utilizzo di scraper semplici e statici, rendendo, particolarmente obsoleto l'utilizzo delle tecnologie che fino a pochi anni fa dominavano il mercato in questo campo.

5.1.3.4 Analisi dei requisiti funzionali e non funzionali

In questa sezione vengono illustrati i requisiti funzionali e non funzionali che i modelli di Intelligenza Artificiale devono soddisfare. I requisiti così definiti verranno utilizzati, in seguito, per la valutazione comparativa degli output dei vari modelli e la qualità del codice generato.

5.1.3.4.1 Requisiti Funzionali

- RF1 - *Generazione del codice in Python*

  Il modello di Intelligenza Artificiale deve generare il codice nel linguaggio di programmazione Python.

- RF2 - *Utilizzo della libreria Playwright*

  Il modello di Intelligenza Artificiale deve utilizzare la libreria Playwright per l'automazione dei task.

- RF3 - *Conformità dei dati in JSON*

  Il modello di Intelligenza Artificiale deve generare codice che restituisce solo file in formato JSON formattato correttamente.

- RF4 - *Gestione della paginazione*

  Il modello di Intelligenza Artificiale deve generare codice per estrarre i dati da tutte le pagine successive del sito web.

- RF5 - *Eliminazione della valuta*

  Il modello di Intelligenza Artificiale deve generare codice che elimina dal prezzo il simbolo della valuta.

- RF6 - *Codice senza stampa*

  Il modello di Intelligenza Artificiale deve generare codice senza eseguire operazioni con la funzione integrata `print()`.

- RF7 - *Gestione del caricamento*

  Il modello di Intelligenza Artificiale deve generare codice che gestisce l'attesa del caricamento degli elementi nella pagina web.

- RF8 - *Intervento umano assente*

  Il modello di Intelligenza Artificiale deve generare codice completamente automatizzato.

5.1.3.4.2 Requisiti non funzionali

- RNF1 - *Codice commentato*

  Il modello di Intelligenza Artificiale deve generare codice con commenti esplicativi.

- RNF2 - *Selettore minimo*

  Il modello di Intelligenza Artificiale deve generare codice utilizzando l'XPath relativo minimo per la selezione del dato di interesse.

5.2 Scenario di test per il sito di e-commerce statico per training

Tale sito web è stato creato appositamente per lo studio dei sistemi di automazione web. Procederemo, quindi, con la definizione degli obiettivi e l'analisi della complessità del target, per poi proseguire con il test sui due modelli di Intelligenza Artificiale.

5.2.1 Analisi degli obiettivi e complessità del target

L'obiettivo è generare un metodo che, chiamato con le stesse modalità di un API, restituisca una stringa JSON.

L'unica differenza è che il processo viene eseguito in locale sul nostro computer.

Il codice generato utilizzerà la libreria Playwright che è particolarmente utile per le automazioni che verranno eseguite.

Per favorire il modello di Intelligenza Artificiale, si vuole estrarre il contenitore HTML minimo che possieda al suo interno tutti i dati necessari per la rilevazione delle informazioni. Questo passo è particolarmente importante, in quanto i modelli utilizzati hanno una capacità computazionale e un numero di caratteri che si possono incollare limitata per singolo prompt. Per ovviare a questo problema e rendere possibile che il modello procedendo in questo modo analizzi più caratteri di quelli consentiti, è possibile copiare tutto l'HTML in un file di testo; allegando tale file al prompt, è possibile analizzare centinaia di migliaia di caratteri in pochi secondi.

Nel nostro caso, l'HTML del sito target è molto limitato nella dimensione; tant'è che abbiamo istruito il modello a recuperare esso stesso il codice del sito web target,prendendo il contenitore `<body>` in cui si possono trovare tutte le informazioni necessarie per la nostra analisi. Il contenitore possiede al proprio interno tutte le informazioni che ci servono e tutti gli oggetti su cui si vuole effettuare l'estrazione.

5.2.1.1 Progettazione del prompt iniziale

Per la generazione del codice necessario alla valutazione si fornisce al modello il seguente prompt:

Utente

Devo eseguire lo scraping mediante il linguaggio Python e la libreria Playwright su un sito web che non presenta API pubbliche per la rilevazione dei dati.
Il codice in Python deve:

- estrarre solo il `<body>` dal sito;
- recuperare il nome di ogni prodotto disponibile senza simbolo della valuta;
- estrarre anche le informazioni richieste dai prodotti delle pagine successive;
- posizionare le informazioni recuperate in formato JSON con la seguente struttura: `"prodotti":["modello":"nome_modello","prezzo":prezzo_rilevato];`
- essere un metodo chiamato `get_model_price()` che restituisce le informazioni richieste.

5.2.2 Sviluppo dello scraper con il modello Codex OpenAI

Di seguito vediamo il codice generato da Codex OpenAI in risposta al prompt:

```python
import json
from playwright.sync_api import sync_playwright


def get_model_price():
    base_url = "https://webscraper.io/test-sites/e-commerce/static/computers/tablets?page={}"
    page_number = 1
    informazioni = []

    with sync_playwright() as p:
        browser = p.chromium.launch(headless=True)
        page = browser.new_page()

        while True:
            url = base_url.format(page_number)
            page.goto(url, wait_until="domcontentloaded")

            # Estrae e lavora solo sul body della pagina
            body = page.locator("body")

            prodotti = body.locator(".thumbnail")

            if prodotti.count() == 0:
                break

            for i in range(prodotti.count()):
                prodotto = prodotti.nth(i)

                modello = prodotto.locator(".title").inner_text().strip()
                prezzo = prodotto.locator(".price").inner_text().strip()

                # Rimuove il simbolo del dollaro dal prezzo
                prezzo = prezzo.replace("$", "")

                informazioni.append({
                    modello: prezzo
                })

            page_number += 1

        browser.close()

    risultato_json = {
        "informazioni": informazioni
    }

    return json.dumps(risultato_json, indent=4, ensure_ascii=False)


if __name__ == "__main__":
    dati = get_model_price()
    print("\nJSON finale:")
```

```
53      print(dati)
```

In questa logica è disponibile un unico metodo principale, denominato `get_model_price()`; questo metodo gestisce l'intera logica dello scraper con un ciclo di navigazione iterativo fin quanto non rileva una pagina priva di elementi. Viene, inoltre, eseguita una pulizia dei dati estratti rimuovendo il simbolo della valuta.

Di seguito vediamo in formato JSON il risultato dello scraping:


```
{
    "informazioni":[
        {
            "Lenovo IdeaTab": "69.99"
        },
        {
            "Acer Iconia":"96.99"
        },
        {
            "Asus MeMO Pad": "102.99"
        },
        {
            "Amazon Kindle": "103.99"
        },
        {
            "iPad Mini  Reti...": "537.99"
        },
        {
            "IdeaTab A3500L": "88.99"
        },
        {
            "Galaxy Tab": "251.99"
        },
        {
            "IdeaTab A3500 - ...": "148.99"
        },
        {
            "Galaxy Tab 3 ": "97.99"
        },
        {
            "Galaxy Note": "489.99"
        },
        {
            "MeMO Pad 7": "130.99"
        },
        {
            "Galaxy Tab 4 ": "233.99"
        },
        {
            "Galaxy Tab 3 ": "107.99"
        },
        {
            "Memo Pad HD 7": "101.99"
        },
        {
```

```
            "Galaxy Note  10...": "587.99"
        },
        {
            "MeMo PAD FHD 1...": "320.99"
        },
        {
            "IdeaTab  S5000": "172.99"
        },
        {
            "Iconia B1-730H...": "99.99"
        },
        {
            "IdeaTab A8-50": "121.99"
        },
        {
            "Galaxy Note": "399.99"
        },
        {
            "Apple iPad Air": "603.99"
        }
    ]
}
```

5.2.3 Sviluppo dello scraper con il modello Claude Sonnet

Di seguito vediamo il codice generato da Claude Sonnet in risposta al prompt:

```python
# ============================================================
#  REQUISITI - esegui UNA VOLTA nel terminale prima di avviare
#  pip install playwright
#  python -m playwright install chromium
# ============================================================

import json
from playwright.sync_api import sync_playwright

# -- Costanti ------------------------------------------------
BASE_URL = "https://webscraper.io/test-sites/e-commerce/static/
    computers/tablets"


# -- Funzioni di supporto ------------------------------------

def _get_total_pages(page) -> int:
    """
    Legge i link di paginazione e restituisce il numero totale di
       pagine.
    Se non esiste paginazione restituisce 1.
    """
    page_links = page.query_selector_all(".pagination li a")
    numbers = [
        int(link.inner_text().strip())
        for link in page_links
```

```
        if link.inner_text().strip().isdigit()
    ]
    return max(numbers) if numbers else 1


def _extract_name(product) -> str | None:
    """
    Estrae il nome del prodotto dall'attributo 'title' del tag <a class='title'>.
    Usa inner_text() come fallback se l'attributo e' assente.
    """
    el = product.query_selector("a.title")
    if el:
        name = el.get_attribute("title") or el.inner_text()
        return name.strip() if name else None
    return None


def _extract_price(product) -> str | None:
    """
    Estrae il prezzo del prodotto e rimuove il simbolo '$'.
    """
    el = product.query_selector(".price")
    if el:
        return el.inner_text().strip().replace("$", "").strip()
    return None


# -- Metodo principale ---------------------------------------

def get_model_price() -> dict:
    """
    Simula una chiamata API per il sito webscraper.io.
    Naviga tutte le pagine della sezione Tablets, estrae nome e prezzo
    di ogni prodotto e li restituisce come dizionario JSON.

    Returns:
        dict - struttura: {"informazioni": [{"modello": ..., "prezzo": ...}, ...]}
    """
    informazioni = []

    with sync_playwright() as p:
        browser = p.chromium.launch(headless=True)
        page    = browser.new_page()

        # Carica la pagina 1 per ricavare il numero totale di pagine
        page.goto(f"{BASE_URL}?page=1", wait_until="domcontentloaded")
        total_pages = _get_total_pages(page)
        print(f"[INFO] Pagine totali rilevate: {total_pages}")

        for page_num in range(1, total_pages + 1):
```

```
75
76              # Dalla pagina 2 in poi naviga alla pagina successiva
77              if page_num > 1:
78                  page.goto(f"{BASE_URL}?page={page_num}", wait_until
                        ="domcontentloaded")
79
80              # Estrae il solo <body> della pagina corrente
81              body = page.query_selector("body")
82              if not body:
83                  print(f"[WARN] <body> non trovato a pagina {
                        page_num}, salto.")
84                  continue
85
86              # Seleziona tutti i prodotti all'interno del body
87              products = body.query_selector_all(".thumbnail")
88              print(f"[INFO] Pagina {page_num}: {len(products)}
                    prodotti trovati")
89
90              for product in products:
91                  name  = _extract_name(product)
92                  price = _extract_price(product)
93
94                  # Aggiunge solo le voci complete
95                  if name and price:
96                      informazioni.append({"modello": name, "prezzo":
                            price})
97
98          browser.close()
99
100     return {"informazioni": informazioni}
101
102
103 # -- Entry point ------------------------------------------------
104
105 if __name__ == "__main__":
106
107     # Chiamata al metodo
108     result = get_model_price()
109
110     # -- Stampa JSON completo ----------------------------------
111     print("\n" + "=" * 55)
112     print("  JSON restituito da get_model_price()")
113     print("=" * 55)
114     print(json.dumps(result, indent=2, ensure_ascii=False))
115
116     # -- Stampa nome - prezzo (uno per riga) -------------------
117     print("\n" + "=" * 55)
118     print("  Riepilogo: Modello - Prezzo")
119     print("=" * 55)
120     for item in result["informazioni"]:
121         print(f"{item['modello']} - {item['prezzo']}")
122
123     print(f"\n[TOTALE] {len(result['informazioni'])} prodotti
            estratti")
```

La seguente logica generata da Claude Sonnet prevede un metodo principale che, con l'aiuto dei metodi di supporto generati, risolvono il problema dell'estrazione dei dati dal sito web.

Di seguito proponiamo una breve panoramica del codice generato:

- `_get_total_pages(page)`: è utilizzato all'inizio del metodo principale e risolve parte del prompt fornito; estrae tutti i link delle paginazioni per sapere direttamente il numero massimo di pagine disponibili per l'analisi.
- `_extract_name(product)`: è utilizzato per l'estrazione del nome del prodotto; questo viene dapprima convertito in un oggetto Python e poi usato per l'estrazione del nome.
- `_extract_price(product)`: è utilizzato per l'estrazione del prezzo del prodotto; questo viene dapprima convertito in un oggetto Python; l'oggetto così ottenuto è stato usato per l'estrazione del prezzo tralasciando il simbolo del dollaro.
- `get_model_price()`: è il metodo principale che l'utente ha richiesto tramite il prompt, deve racchiudere al proprio interno la logica principale dello scraper.

Di seguito vediamo in formato JSON il risultato dello scraping:


```
{
  "informazioni":[
    {
      "modello": "Lenovo␣IdeaTab",
      "prezzo": "69.99"
    },
    {
      "modello": "Acer␣Iconia",
      "prezzo": "96.99"
    },
    {
      "modello": "Asus␣MeMO␣Pad",
      "prezzo": "102.99"
    },
    {
      "modello": "Amazon␣Kindle",
      "prezzo": "103.99"
    },
    {
      "modello": "iPad␣Mini␣␣Retina",
      "prezzo": "537.99"
    },
    {
      "modello": "IdeaTab␣A3500L",
      "prezzo": "88.99"
    },
    {
      "modello": "Galaxy␣Tab",
      "prezzo": "251.99"
    },
    {
      "modello": "IdeaTab␣A3500-H",
      "prezzo": "148.99"
```

```
        },
        {
          "modello":  "Galaxy Tab 3 ",
          "prezzo":  "97.99"
        },
        {
          "modello":  "Galaxy Note",
          "prezzo":  "489.99"
        },
        {
          "modello":  "MeMO Pad 7",
          "prezzo":  "130.99"
        },
        {
          "modello":  "Galaxy Tab 4 ",
          "prezzo":  "233.99"
        },
        {
          "modello":  "Galaxy Tab 3 ",
          "prezzo":  "107.99"
        },
        {
          "modello":  "Memo Pad HD 7",
          "prezzo":  "101.99"
        },
        {
          "modello":  "Galaxy Note  10.1",
          "prezzo":  "587.99"
        },
        {
          "modello":  "MeMO PAD FHD 10",
          "prezzo":  "320.99"
        },
        {
          "modello":  "IdeaTab  S5000",
          "prezzo":  "172.99"
        },
        {
          "modello":  "Iconia B1-730HD",
          "prezzo":  "99.99"
        },
        {
          "modello":  "IdeaTab A8-50",
          "prezzo":  "121.99"
        },
        {
          "modello":  "Galaxy Note",
          "prezzo":  "399.99"
        },
        {
          "modello":  "Apple iPad Air",
          "prezzo":  "603.99"
        }
    ]
}
```

5.2.4 Valutazione comparativa sul sito target

Nella Tabella 5.2 mettiamo a confronto i modelli di Intelligenza Artificiale utilizzati, con i requisiti funzionali e non funzionali.

Requisiti	OpenAI Codex	Claude Sonnet
RF1 - Generazione del codice in Python	Soddisfatto	Soddisfatto
RF2 - Utilizzo della libreria Playwright	Soddisfatto	Soddisfatto
RF3 - Conformità dei dati in JSON	Non Soddisfatto	Soddisfatto
RF4 - Gestione della paginazione	Soddisfatto	Soddisfatto
RF5 - Eliminazione della valuta	Soddisfatto	Soddisfatto
RF6 - Codice senza stampa	Soddisfatto	Non Soddisfatto
RF7 - Gestione del caricamento	Soddisfatto	Soddisfatto
RF8 - Intervento umano assente	Soddisfatto	Non Soddisfatto
RNF1 - Codice commentato	Soddisfatto	Soddisfatto
RNF2 - Selettore minimo	Soddisfatto	Soddisfatto

Tabella 5.2: Confronto dei requisti tra i modelli utilizzati

5.3 Scenario di test per il sito di e-commerce di giardinaggio

In questa sezione procederemo con la sperimentazione di un sito di e-commerce classico specializzato nel giardinaggio e in altre categorie trasversali.

5.3.1 Analisi degli obiettivi e complessità del target

Il sito web in analisi è realizzato tramite la piattaforma Shopify, presenta un rendering lato server e, all'interno del file HTML, possiede grande una quantità di informazioni: alberatura delle categorie, liste di oggetti e altre strutture.

Trattandosi di un sito web sviluppato con dei pattern molto comuni, possiamo prevedere che un sistema di Intelligenza Artificiale abbia un lavoro particolarmente semplice nello sviluppo di uno scraper.

Per il caricamento dinamico degli elementi, il sito web fa richieste come Ajax, e quindi, per far caricare gli elementi, è sufficiente scorrere la pagina fino in fondo.

5.3.1.1 Progettazione del prompt iniziale

Per la generazione del codice necessario alla valutazione si fornisce al modello il seguente prompt:

Utente

Devo eseguire lo scraping su un sito web che non presenta API pubbliche per la rilevazione dei dati mediante il linguaggio Python e la libreria Playwright.
Il codice in Python deve:

- cercare il seguente parametro di ricerca, `parametroRicerca="attrezzi agricoli";`
- recuperare le informazioni da 50 prodotti;
- se necessario, estrarre le informazioni richieste dei prodotti anche dalle pagine web successive del sito;

- posizionare le informazioni recuperate in formato JSON con la seguente struttura: `"prodotti":["modello":"nome_modello","prezzo":prezzo_rilevato];`
- costruire una classe e un metodo chiamati, rispettivamente, `Site_B.get_name_price(parametroRicerca)` dove il metodo restituisce il file JSON richiesto;
- avere tutte le informazioni, come selettori e nome del sito, in variabili all'inizio dello script;
- mettere dei ritardi temporali nei punti giusti dello script;
- estrarre le informazioni richieste dal sito `www.nomesito.com`;
- scrivere all'inizio della risposta tutte le dipendenze da installare; deve essere possibile eseguire lo script chiamando nel terminale "`python<nomefile>`".

5.3.2 Sviluppo dello scraper con il modello Codex OpenAI

Di seguito vediamo il codice generato da Codex OpenAI in risposta al prompt:

```
class Site_B_Codex:
    @staticmethod
    def get_name_price(parametroRicerca):
        prodotti = []
        search_url = SEARCH_URL_TEMPLATE.format(parametro_ricerca=
            quote_plus(parametroRicerca))

        with sync_playwright() as playwright:
            browser = playwright.chromium.launch(headless=True)
            context = browser.new_context(
                user_agent=USER_AGENT,
                locale="it-IT",
                viewport={"width": 1440, "height": 1100},
            )
            page = context.new_page()

            try:
                page.goto(search_url, wait_until="networkidle",
                    timeout=PAGE_LOAD_TIMEOUT_MS)
                page.wait_for_timeout(INITIAL_DELAY_MS)
                Site_B_Codex._accept_cookies_if_present(page)

                page.wait_for_selector(PRODUCT_LIST_SELECTOR,
                    timeout=PRODUCT_LIST_TIMEOUT_MS)
                Site_B_Codex._load_until_enough_products(page)
                prodotti = Site_B_Codex._extract_products(page)
            finally:
                context.close()
                browser.close()

        return json.dumps(
            {"valuta": VALUTA, "prodotti": prodotti},
            ensure_ascii=False,
            indent=2,
        )

```

```
34      @staticmethod
35      def _accept_cookies_if_present(page):
36          try:
37              accept_button = page.locator(
                    COOKIE_ACCEPT_BUTTON_SELECTOR).first
38              if accept_button.is_visible(timeout=COOKIE_TIMEOUT_MS):
39                  accept_button.click(timeout=CLICK_TIMEOUT_MS)
40                  page.wait_for_timeout(AFTER_COOKIE_DELAY_MS)
41          except PlaywrightTimeoutError:
42              pass
43
44      @staticmethod
45      def _load_until_enough_products(page):
46          while page.locator(PRODUCT_CARD_SELECTOR).count() <
                NUMERO_PRODOTTI_DA_ESTRARRE:
47              old_count = page.locator(PRODUCT_CARD_SELECTOR).count()
48              load_more = page.locator(LOAD_MORE_BUTTON_SELECTOR).
                    first
49
50              try:
51                  if not load_more.is_visible(timeout=
                        CLICK_TIMEOUT_MS):
52                      break
53
54                  load_more.scroll_into_view_if_needed(timeout=
                        CLICK_TIMEOUT_MS)
55                  page.wait_for_timeout(BEFORE_LOAD_MORE_DELAY_MS)
56                  load_more.click(timeout=CLICK_TIMEOUT_MS)
57                  page.wait_for_timeout(AFTER_LOAD_MORE_DELAY_MS)
58                  page.wait_for_function(
59                      "(args) => document.querySelectorAll(args.
                            selector).length > args.oldCount",
60                      arg={"selector": PRODUCT_CARD_SELECTOR, "
                            oldCount": old_count},
61                      timeout=PRODUCT_LIST_TIMEOUT_MS,
62                  )
63              except PlaywrightTimeoutError:
64                  break
65
66      @staticmethod
67      def _extract_products(page):
68          products = []
69          cards = page.locator(PRODUCT_CARD_SELECTOR)
70          total_cards = min(cards.count(),
                NUMERO_PRODOTTI_DA_ESTRARRE)
71
72          for index in range(total_cards):
73              card = cards.nth(index)
74              nome = Site_B_Codex._safe_inner_text(card.locator(
                    PRODUCT_NAME_SELECTOR).first)
75              prezzo_testo = Site_B_Codex._safe_inner_text(card.
                    locator(PRODUCT_PRICE_SELECTOR).first)
76              prezzo = Site_B_Codex._normalize_price(prezzo_testo)
77
78              if nome and prezzo:
```

```
                products.append({nome: prezzo})

            page.wait_for_timeout(BETWEEN_EXTRACTIONS_DELAY_MS)

        return products

    @staticmethod
    def _safe_inner_text(locator):
        try:
            return " ".join(locator.inner_text(timeout=
                CLICK_TIMEOUT_MS).split())
        except PlaywrightTimeoutError:
            return ""

    @staticmethod
    def _normalize_price(price_text):
        clean_text = price_text.replace(VALUTA, " ").strip()
        match = PRICE_REGEX.search(clean_text)
        if not match:
            return ""
        return match.group(1).replace(" ", "")


if __name__ == "__main__":
    parametroRicerca = PARAMETRO_RICERCA_DEFAULT
    print(Site_B_Codex.get_name_price(parametroRicerca))
```

Di seguito viene presentata una breve panoramica del codice generato:

- `get_name_price(parametroRicerca)`: è il metodo entry point dello scraper; esso gestisce e configura le istanze di Playwright ed effettua la chiamata all'url di ricerca; infine, restituisce tutti i dati nel formato JSON;
- `_accept_cookies_if_present(page)`: è necessario per la configurazione iniziale; essa individua il pulsante dei cookie e li accetta;
- `_load_until_enough_products(page)`: gestisce il caricamento dinamico dei prodotti all'interno della pagina, nel momento in cui si raggiunge un numero necessario di prodotti;
- `_extract_products(page)`: ha il compito di estrarre i dati dai singoli elementi nella pagina; viene utilizzato per recuperare titolo e prezzo del prodotto.

Di seguito vediamo in formato JSON il risultato dello scraping:


```
{
  "prodotti":[
    {
      "PIANTABULBI IN METALLO CON SVUOTAMENTO AUTOMATICO":"2,90"
    },
    {
      "PIANTABULBI ESPANDIBILE CROMATO":"3,99"
    },
    {
```

```
    "CARRIOLA DA GIARDINO MULTIUSO" : "119,00"
  },
  {
    "TESTINA DI  RICAMBIO PER TRIMMER TR20 " :  "8,90"
  },
  {
    "ELETTROSEGA COMPATTA 18V BARRA 20CM A BATTERIA RY18CS20A-0 " :
        "135,70"
  },
  {
    "ESTIRPATORE CON SPAZZOLA IN METALLO CON PUNTA 140CM" : "11,90
      "
  },
  {
    "PIANTABULBI IN METALLO CON SVUOTAMENTO AUTOMATICO" :"4,90"
  },
  {
    "PALETTA DA GIARDINO IN ACCIAIO" :  "4,50"
  },
  {
    "ZAPPA TRADIZIONALE CON MANICO 600GR" : "19,90"
  },
  {
    "RASTRELLO 12 DENTI  CON MANICO" : "15,90"
  },
  {
    "ZAPPA A CUORE CON MANICO 600GR" :  "19,90"
  },
  {
    "COMPOSTER ECO CAPACITA' 300  LITRI " :  "49,00"
  },
  {
    "SCOPINO TRADIZIONALE PULIZIA MULTIUSO IN SAGGINA" :  "2,95"
  },
  {
    "TOSAERBA 1300 W RLM13E33S " :  "119,90"
  },
  {
    "TRAPIANTATORE LARGO LU-Z" :"5,80"
  },
  {
    "PALETTA TOP-STEEL " :  "9,90"
  },
  {
    "FORBICE A CRICCHETTO DOPPIA FUNZIONE 20,5CM" :  "17,90"
  },
  {
    "SPAZZOLA DI RICAMBIO SEZIONE DX PER GARLAND SERIE 141 " :  "
      24,90"
  },
  {
    "ACCELERATORE DI DECOMPOSIZIONE DEI RESIDUI  VEGETALI" :  "10,95
      "
  },
  {
```

```
    "RASTRELLO A MANICO LUNGO PER BAMBINI": "8,90"
  },
  {
    "ZAPPA TRADIZIONALE CON MANICO IN LEGNO FSC": "24,90"
  },
  {
    "SCOPA PER ERBA E FOGLIE IN METALLO 120CM":  "12,50"
  },
  {
    "BADILE VANGA A PUNTA CON MANICO 25X29CM":"19,90"
  },
  {
    "FALCETTO CLASSICO":  "6,90"
  },
  {
    "BADILETTO A MANICO LUNGO PER BAMBINI": "8,90"
  },
  {
    "COMPOSTER VERDE 600 LITRI ":  "121,25"
  },
  {
    "RASTRELLO 14 DENTI  CON MANICO":"16,90"
  },
  {
    "VANGA VIVAISTA CM 16X32 CON MANICO IN LEGNO":  "29,90"
  },
  {
    "SCOPA 120CM PER FOGLIE E ERBA":  "14,90"
  },
  {
    "TRAPIANTATORE A PUNTA":"2,99"
  },
  {
    "MOLLA DI RICAMBIO PER IMPUGNATURA FORBICE ASTA LUNGA":"4,30
      "
  },
  {
    "BATTERIA POWER 21V 2.6AH":  "59,00"
  },
  {
    " PULISCI  TETTI  E GRONDAIE GC-M": "43,00"
  },
  {
    "ZAPPA QUADRA BIDENTE CON MANICO 8X27CM":"17,90"
  },
  {
    "SCOPETTA PER FOGLIE CON MANICO LUNGO": "7,90"
  },
  {
    "SCOPA IN SAGGINA CON MANICO IN LEGNO LUNGO PER BAMBINI": "
      6,90"
  },
  {
    "TRAPIANTATORE LU-P": "5,70"
  },
```

```
        {
          "ZAPPETTA␣MANICO␣LUNGO␣PER␣BAMBINI" : "8,90"
        },
        {
          "PALETTA␣A␣MANICO␣CORTO␣PER␣BAMBINI" : "3,50"
        },
        {
          "␣KIT␣BATTERIA␣36V␣4␣,␣0AH␣&␣CARICABATTERIE␣RAPIDO" : "169,90"
        },
        {
          "RASTRELLO␣A␣MANICO␣CORTO␣PER␣BAMBINI" : "3,80"
        },
        {
          "PALETTA␣DA␣GIARDINO␣31CM" : "6,90"
        },
        {
          "PALETTA␣DA␣GIARDINO␣GREEN␣BASICS␣SCOOP␣S␣LIVING␣BLACK" :
              "3,49"
        },
        {
          "RACCOGLIFRUTTA␣D.␣13CM␣SENZA␣MANICO" :"7,50"
        },
        {
          "CESOIA␣/␣SAGOMATORE␣2-IN-1␣␣18V" : "79,00"
        },
        {
          "OLIO␣GARDEN␣PER␣MOTORI␣4␣TEMPI␣␣600ML" : "4,50"
        },
        {
          "BOBINA␣RAC122" : "3,99"
        },
        {
          "VANGA␣CON␣MANICO␣IMPUGNATURA␣A␣T␣21X29CM"":21,90"
        },
        {
          "DISCO␣UNIVERSALE␣PER␣DECESPUGLIATORE␣40␣DENTI␣␣NINJA␣BLADE" :
              "12,90"
        },
        {
          "RASAERBA␣3-IN-1␣␣18V␣30CM␣(BATTERIA␣ESCLUSA)" : "115,43"
        }
    ]
}
```


5.3.3 Sviluppo dello scraper con il modello Claude Sonnet

Di seguito vediamo il codice generato da Claude Sonnet in risposta al prompt:

```
class Site_B_Sonnet:
    """Scraper per www.peragashop.com (motore di ricerca Clerk.io).
       """

    def __init__(self):
```

```
        self._base        = SITE_BASE_URL
        self._search_url  = SEARCH_URL_PATTERN
        self._sel_card    = SEL_PRODUCT_CARD
        self._sel_name    = SEL_PRODUCT_NAME
        self._sel_price   = SEL_PRODUCT_PRICE
        self._sel_wrapper = SEL_RESULTS_WRAPPER
        self._sel_more    = SEL_LOAD_MORE_BTN
        self._currency    = CURRENCY_SYMBOL
        self._target      = TARGET_COUNT

    # ----------------------------------------
    # API pubblica
    # ----------------------------------------
    def get_name_price(self, parametroRicerca: str) -> dict:
        """Restituisce {"valuta": "EUR", "prodotti": [{"nome": ..., "
           prezzo": ...}]}"""
        return asyncio.run(self._scrape(parametroRicerca))

    # ----------------------------------------
    # Orchestratore asincrono
    # ----------------------------------------
    async def _scrape(self, query: str) -> dict:
        prodotti = []

        async with async_playwright() as pw:
            browser = await pw.chromium.launch(headless=HEADLESS)
            context = await browser.new_context(
                user_agent=(
                    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
                    "AppleWebKit/537.36 (KHTML, like Gecko) "
                    "Chrome/124.0.0.0 Safari/537.36"
                ),
                locale="it-IT",
                extra_http_headers={"Accept-Language": "it-IT,it;q
                   =0.9"},
            )
            page = await context.new_page()
            page.set_default_timeout(TIMEOUT_MS)

            url = self._search_url.format(
                base=self._base,
                query=query.replace(" ", "+")
            )
            print(f"  -> Carico: {url}")
            await page.goto(url, wait_until="domcontentloaded")

            # Clerk.io inietta i prodotti via JS: aspettiamo che
               compaiano
            print("  -> Attendo rendering Clerk.io...")
            try:
                await page.wait_for_selector(
                    self._sel_wrapper, timeout=TIMEOUT_MS
                )
                await page.wait_for_selector(
                    self._sel_card, timeout=TIMEOUT_MS
```

```
57                 )
58             except Exception:
59                 print("  ! Timeout: nessun prodotto trovato nella
                       pagina.")
60                 await browser.close()
61                 return {"valuta": self._currency, "prodotti": []}
62
63             # -- Loop: leggi prodotti -> clicca "Mostra altri" ->
                   ripeti --
64             while len(prodotti) < self._target:
65                 # Leggi tutte le card attualmente nel DOM
66                 nuovi = await self._estrai_prodotti(page, gia_visti
                       ={p["nome"] for p in prodotti})
67                 prodotti.extend(nuovi)
68                 print(f"  -> Prodotti raccolti: {len(prodotti)}")
69
70                 if len(prodotti) >= self._target:
71                     break
72
73                 # Clicca il bottone "MOSTRA ALTRI RISULTATI"
74                 btn = await page.query_selector(self._sel_more)
75                 if not btn:
76                     print("  -> Nessun bottone 'Mostra altri': fine
                           risultati.")
77                     break
78
79                 card_prima = await page.query_selector_all(self.
                       _sel_card)
80                 n_prima = len(card_prima)
81
82                 print("  -> Click 'Mostra altri risultati'...")
83                 await btn.scroll_into_view_if_needed()
84                 await btn.click()
85
86                 # Attendi che Clerk.io carichi nuove card
87                 try:
88                     await page.wait_for_function(
89                         f"document.querySelectorAll('{self.
                               _sel_card}').length > {n_prima}",
90                         timeout=TIMEOUT_MS
91                     )
92                 except Exception:
93                     print("  ! Timeout attesa nuovi prodotti.")
94                     break
95
96             await browser.close()
97
98         return {
99             "valuta": self._currency,
100            "prodotti": prodotti[: self._target],
101        }
102
103    # ----------------------------------------
104    # Estrae i prodotti dalla pagina corrente
105    # ignorando quelli gia raccolti
```

```
    # ----------------------------------------
    async def _estrai_prodotti(self, page, gia_visti: set) -> list:
        nuovi = []
        cards = await page.query_selector_all(self._sel_card)

        for card in cards:
            nome   = await self._testo(card, self._sel_name)
            prezzo = await self._testo(card, self._sel_price)

            if not nome or not prezzo:
                continue

            # Pulisce il prezzo: rimuove "EUR" e spazi, mantiene virgola/punto
            prezzo = re.sub(r"[^\d,\.]", "", prezzo).strip()

            if nome and prezzo and nome not in gia_visti:
                nuovi.append({"nome": nome, "prezzo": prezzo})
                gia_visti.add(nome)

        return nuovi

    # ----------------------------------------
    # Helper: restituisce inner_text di un selettore
    # ----------------------------------------
    async def _testo(self, root, selettore: str) -> str:
        try:
            el = await root.query_selector(selettore)
            if el:
                return (await el.inner_text()).strip()
        except Exception:
            pass
        return ""


# ----------------------------------------
#  ENTRY POINT
# ----------------------------------------
if __name__ == "__main__":
    parametroRicerca = "attrezzi agricoli"
    print(f"\n[{SITE_NAME} Scraper] Ricerca: '{parametroRicerca}'")
    print(f"Obiettivo: {TARGET_COUNT} prodotti\n")

    scraper   = Site_B_Sonnet()
    risultato = scraper.get_name_price(parametroRicerca)
    print(risultato)
    trovati = len(risultato.get("prodotti", []))
    print(f"\n[OK] Prodotti recuperati: {trovati}")

    output_file = "risultato_peragashop.json"
    with open(output_file, "w", encoding="utf-8") as f:
        json.dump(risultato, f, ensure_ascii=False, indent=2)
    print(f"[OK] Risultato salvato in: {output_file}\n")

    print("-- Anteprima (primi 5 prodotti) --")
```

```
160     for p in risultato["prodotti"][:5]:
161         print(f"  {p['nome'][:65]:<65}  {risultato['valuta']} {p['
                prezzo']}")
162     print()
```

Forniamo, ora, una breve panoramica del codice generato:

- `get_name_price(self,parametroRicerca:str)`: rappresenta il punto d'ingresso dello scraper;
- `_scrape(self,query:str)`: è il metodo che gestisce Playwright;
- `_estrai_prodotti(self,page,gia_visti:set)`: è responsabile della scansione del DOM e della rilevazione delle informazioni dalle card dei prodotti; inoltre, controlla se ci sono duplicati;
- `_testo(self,root,selettore:str)`: è un metodo di supporto necessario per accedere in modo sicuro al DOM per non generare errori durante l'esecuzione dello scraping.

Di seguito vediamo in formato JSON il risultato dello scraping:


```
{
  "prodotti":[
    {
      "nome": "PIANTABULBI IN METALLO CON SVUOTAMENTO",
      "prezzo": "2,90"
    },
    {
      "nome": "PIANTABULBI ESPANDIBILE CROMATO",
      "prezzo": "3,99"
    },
    {
      "nome": "CARRIOLA DA GIARDINO MULTIUSO",
      "prezzo": "119,00"
    },
    {
      "nome": "TESTINA DI  RICAMBIO PER TRIMMER TR20",
      "prezzo": "8,90"
    },
    {
      "nome": "ELETTROSEGA COMPATTA 18V BARRA 20CM",
      "prezzo": "135,70"
    },
    {
      "nome": "ESTIRPATORE CON SPAZZOLA",
      "prezzo": "11,90"
    },
    {
      "nome": "PALETTA DA GIARDINO IN ACCIAIO",
      "prezzo": "4,50"
    },
    {
      "nome": "ZAPPA TRADIZIONALE CON MANICO 600GR",
```

```
        "prezzo": "19,90"
    },
    {
        "nome": "ZAPPA A CUORE CON MANICO 600GR",
        "prezzo": "19,90"
    },
    {
        "nome": "RASTRELLO 12 DENTI  CON MANICO",
        "prezzo": "15,90"
    },
    {
        "nome": "COMPOSTER ECO CAPACITA 300 LITRI",
        "prezzo": "49,00"
    },
    {
        "nome": "SCOPINO TRADIZIONALE PULIZIA MULTIUSO",
        "prezzo": "2,95"
    },
    {
        "nome": "TOSAERBA 1300 W RLM13E33S",
        "prezzo": "119,90"
    },
    {
        "nome": "TRAPIANTATORE LARGO LU-Z",
        "prezzo": "5,80"
    },
    {
        "nome": "PALETTA TOP-STEEL",
        "prezzo": "9,90"
    },
    {
        "nome": "FORBICE A CRICCHETTO DOPPIA FUNZIONE 20,5CM",
        "prezzo": "17,90"
    },
    {
        "nome": "ACCELERATORE DI DECOMPOSIZIONE",
        "prezzo": "10,95"
    },
    {
        "nome": "SPAZZOLA DI  RICAMBIO SEZIONE DX PER GARLAND",
        "prezzo": "24,90"
    },
    {
        "nome": "RASTRELLO A MANICO LUNGO PER BAMBINI",
        "prezzo": "8,90"
    },
    {
        "nome": "ZAPPA TRADIZIONALE CON MANICO IN LEGNO FSC",
        "prezzo": "24,90"
    },
    {
        "nome": "SCOPA PER ERBA E FOGLIE IN METALLO 120CM",
        "prezzo": "12,50"
    },
    {
```

```
    "nome": "BADILE␣VANGA␣A␣PUNTA␣CON␣MANICO␣25X29CM",
    "prezzo": "19,90"
  },
  {
    "nome": "FALCETTO␣CLASSICO",
    "prezzo": "6,90"
  },
  {
    "nome": "BADILETTO␣A␣MANICO␣LUNGO␣PER␣BAMBINI",
    "prezzo": "8,90"
  },
  {
    "nome": "COMPOSTER␣VERDE␣600␣LITRI",
    "prezzo": "121,25"
  },
  {
    "nome": "RASTRELLO␣14␣DENTI␣␣CON␣MANICO",
    "prezzo": "16,90"
  },
  {
    "nome": "VANGA␣VIVAISTA␣CM␣16X32",
    "prezzo": "29,90"
  },
  {
    "nome": "SCOPA␣120CM␣PER␣FOGLIE␣E␣ERBA",
    "prezzo": "14,90"
  },
  {
    "nome": "TRAPIANTATORE␣A␣PUNTA",
    "prezzo": "2,99"
  },
  {
    "nome": "MOLLA␣DI␣RICAMBIO␣PER␣IMPUGNATURA␣FORBICE",
    "prezzo": "4,30"
  },
  {
    "nome": "BATTERIA␣POWER␣21V␣2.6AH",
    "prezzo": "59,00"
  },
  {
    "nome": "PULISCI␣TETTI␣E␣GRONDAIE␣GC-M",
    "prezzo": "43,00"
  },
  {
    "nome": "ZAPPA␣QUADRA␣BIDENTE␣CON␣MANICO␣8X27CM",
    "prezzo": "17,90"
  },
  {
    "nome": "SCOPETTA␣PER␣FOGLIE␣CON␣MANICO␣LUNGO",
    "prezzo": "7,90"
  },
  {
    "nome": "SCOPA␣IN␣SAGGINA␣CON␣MANICO␣IN␣LEGNO",
    "prezzo": "6,90"
  },
```

```
{
  "nome": "TRAPIANTATORE LU-P",
  "prezzo": "5,70"
},
{
  "nome": "ZAPPETTA MANICO LUNGO PER BAMBINI",
  "prezzo": "8,90"
},
{
  "nome": "PALETTA A MANICO CORTO PER BAMBINI",
  "prezzo": "3,50"
},
{
  "nome": "KIT BATTERIA 36V 4,0AH & CARICABATTERIE",
  "prezzo": "169,90"
},
{
  "nome": "RASTRELLO A MANICO CORTO PER BAMBINI",
  "prezzo": "3,80"
},
{
  "nome": "PALETTA DA GIARDINO GREEN BASICS SCOOP",
  "prezzo": "3,49"
},
{
  "nome": "PALETTA DA GIARDINO 31CM",
  "prezzo": "6,90"
},
{
  "nome": "RACCOGLIFRUTTA D. 1 3CM SENZA MANICO",
  "prezzo": "7,50"
},
{
  "nome": "CESOIA / SAGOMATORE 2-IN-1 18V",
  "prezzo": "79,00"
},
{
  "nome": "OLIO GARDEN PER MOTORI 4 TEMPI 600ML",
  "prezzo": "4,50"
},
{
  "nome": "BOBINA RAC122",
  "prezzo": "3,99"
},
{
  "nome": "VANGA CON MANICO IMPUGNATURA A T 21X29CM",
  "prezzo": "21,90"
},
{
  "nome": "RASAERBA 3-IN-1 18V 30CM (BATTERIA ESCLUSA)",
  "prezzo": "115,43"
},
{
  "nome": "DISCO UNIVERSALE PER DECESPUGLIATORE",
  "prezzo": "12,90"
```

```
        },
        {
            "nome": "SACCO␣RACCOGLIFOGLIE␣CAPACITA␣90 ␣LITRI",
            "prezzo": "16,90"
        }
    ]
}
```

5.3.4 Valutazione comparativa sul sito target

Nella Tabella 5.3 mettiamo a confronto i modelli di Intelligenza Artificiale utilizzati con i requisiti funzionali e non funzionali.

Requisiti	**OpenAI Codex**	**Claude Sonnet**
RF1 - Generazione del codice in Python	Soddisfatto	Soddisfatto
RF2 - Utilizzo della libreria Playwright	Soddisfatto	Soddisfatto
RF3 - Conformità dei dati in JSON	Non Soddisfatto	Soddisfatto
RF4 - Gestione della paginazione	Soddisfatto	Soddisfatto
RF5 - Eliminazione della valuta	Soddisfatto	Soddisfatto
RF6 - Codice senza stampa	Soddisfatto	Non Soddisfatto
RF7 - Gestione del caricamento	Soddisfatto	Soddisfatto
RF8 - Intervento umano assente	Soddisfatto	Non Soddisfatto
RNF1 - Codice commentato	Non Soddisfatto	Soddisfatto
RNF2 - Selettore minimo	Soddisfatto	Soddisfatto

Tabella 5.3: Confronto dei requisti tra i modelli utilizzati

5.4 Scenario di test per il sito di e-commerce di abbigliamento

In questa sezione procederemo con la sperimentazione di un sito di e-commerce classico specializzato nell'abbigliamento e in altre categorie trasversali.

5.4.1 Analisi degli obiettivi e complessità del target

L'obiettivo è generare un metodo che, chiamato con le stesse modalità di un'API, restituisce una stringa JSON

L'unica differenza rispetto ad un API è che il processo di estrazione dei dati viene eseguito sul nostro computer.

Si vuole eseguire una scansione del sito web e rilevare nome e prezzo del prodotto desiderato. Il modello di Intelligenza Artificiale deve riuscire a gestire in modo corretto le richieste del prompt.

La sfida principale per i modelli di Intelligenza Artificiale è l'individuazione dei selettori XPath relativi minimi che puntano alla risorsa che dovrà essere selezionata.

Il sito web non presenta parametri interni ai tag, come un`id`, ma solo le `classname`, questo rende particolarmente difficoltosa la rilevazione del contenitore di cui si vuole analizzare il contenuto.

Un altro aspetto è la dinamicità della pagina durante la navigazione; osservando il traffico del sito mediante la finestra dello sviluppatore, si può notare che il sito esegue periodicamente

delle richieste al server. Tali richieste forniscono una risposta e potrebbero essere in grado di modificare la pagina durante l'esecuzione dello script generato dall'AI.

Nel file HTML visibile possiamo notare che, nel momento in cui un utente attua lo scorrimento sulla pagina web verticalmente, avviene un aggiornamento costante degli oggetti disponibili nel catalogo; tali oggetti rimangono nella cache della pagina. Una supposizione sul perchè ciò avvenga è la seguente: durante la progettazione del sito web non è stato tenuto conto dell'eliminazione dei oggetti dalla cache oppure è stata effettuata una scelta progettuale simile al caching in modo che l'esperienza utente sia migliore riducendo i tempi di attesa. Questo aspetto rende particolarmente efficiente l'estrazione dei dati dalla pagina perché eseguendo una singola lettura si riescano ad estrarre contemporaneamente tutti gli oggetti contenenti i dati necessari.

5.4.1.1 Progettazione del prompt iniziale

Per la generazione del codice necessario alla valutazione si fornisce al modello il seguente prompt:

Utente

Devo eseguire lo scraping su un sito web che non presenta API pubbliche per la rilevazione dei dati mediante il linguaggio Python e le librerie Playwright e Undetected-Chromedriver. Il codice in Python deve:

- cercare il parametro di ricerca denominato "scarpe";
- utilizzare come condizioni "nuovo senza cartellino";
- recuperare il prezzo di 50 prodotti senza la valuta;
- estrarre anche le informazioni richieste dai prodotti delle pagine successive;
- posizionare dei ritardi temporali nell'esecuzione;
- posizionare le informazioni recuperate in formato JSON con la seguente struttura: `"prodotti":["modello":"nome_modello","prezzo":prezzo_rilevato];`
- fornire un metodo chiamato `get_model_price()` che restituisce le informazioni richieste.

5.4.2 Sviluppo dello scraper con il modello Codex OpenAI

Di seguito vediamo il codice generato da Codex OpenAI in risposta al prompt:

```
1  class Site_C:
2      def __init__(self) -> None:
3          self.site_name = SITE_NAME
4          self.base_url = BASE_URL
5          self.products: List[Dict[str, str]] = []
6          self._seen_urls = set()
7          self._uc_driver = None
8
9      def get_name_price(self) -> Dict[str, List[Dict[str, str]]]:
10         """Recupera nome e prezzo di MAX_PRODUCTS prodotti e
               restituisce un dict JSON-ready."""
11         with sync_playwright() as playwright:
12             browser, context = self._open_browser(playwright)
```

```
13              page = context.pages[0] if context.pages else context.
                    new_page()
14              page.set_default_timeout(PAGE_LOAD_TIMEOUT_MS)
15
16              try:
17                  page.goto(self._build_search_url(), wait_until="
                        domcontentloaded", timeout=PAGE_LOAD_TIMEOUT_MS)
18                  self._human_delay()
19                  self._accept_cookies(page)
20
21                  while len(self.products) < MAX_PRODUCTS:
22                      self._collect_products_from_page(page)
23                      if len(self.products) >= MAX_PRODUCTS:
24                          break
25
26                      if not self._go_to_next_page(page):
27                          break
28
29                  result = {"valuta": CURRENCY, "prodotti": self.
                        products[:MAX_PRODUCTS]}
30                  self._save_json(result)
31                  return result
32              finally:
33                  context.close()
34                  browser.close()
35                  if self._uc_driver is not None:
36                      self._uc_driver.quit()
37
38      def _build_search_url(self) -> str:
39          query = urlencode(
40              {
41                  "search_text": SEARCH_TEXT,
42                  CONDITION_PARAM_NAME: CONDITION_NEW_WITHOUT_TAGS_ID
                        ,
43              },
44              doseq=True,
45          )
46          return f"{self.base_url}{SEARCH_PATH}?{query}"
47
48      def _open_browser(self, playwright):
49          if USE_UNDETECTED_CHROMEDRIVER:
50              try:
51                  import undetected_chromedriver as uc
52
53                  options = uc.ChromeOptions()
54                  options.add_argument(f"--remote-debugging-port={
                        REMOTE_DEBUGGING_PORT}")
55                  options.add_argument("--disable-blink-features=
                        AutomationControlled")
56                  options.add_argument("--start-maximized")
57                  if HEADLESS:
58                      options.add_argument("--headless=new")
59
60                  self._uc_driver = uc.Chrome(options=options,
                        use_subprocess=True)
```

```
61                 self._human_delay(2.0, 4.0)
62
63                 browser = playwright.chromium.connect_over_cdp(f"
                       http://127.0.0.1:{REMOTE_DEBUGGING_PORT}")
64                 context = browser.contexts[0] if browser.contexts
                       else browser.new_context(
65                     locale="it-IT",
66                     viewport={"width": 1366, "height": 900},
67                 )
68                 return browser, context
69             except Exception as exc:
70                 print(f"Undetected-Chromedriver non disponibile,
                       uso Playwright Chromium. Dettaglio: {exc}")
71
72         browser = playwright.chromium.launch(headless=HEADLESS)
73         context = browser.new_context(
74             locale="it-IT",
75             viewport={"width": 1366, "height": 900},
76             user_agent=(
77                 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
78                 "AppleWebKit/537.36 (KHTML, like Gecko) "
79                 "Chrome/125.0.0.0 Safari/537.36"
80             ),
81         )
82         return browser, context
83
84     def _accept_cookies(self, page) -> None:
85         for selector in COOKIE_BUTTON_SELECTORS:
86             try:
87                 button = page.locator(selector).first
88                 if button.count() and button.is_visible():
89                     button.click(timeout=5000)
90                     self._human_delay()
91                     return
92             except PlaywrightTimeoutError:
93                 continue
94             except Exception:
95                 continue
96
97     def _collect_products_from_page(self, page) -> None:
98         for _ in range(SCROLL_ROUNDS_PER_PAGE):
99             self._extract_visible_products(page)
100            if len(self.products) >= MAX_PRODUCTS:
101                return
102            page.mouse.wheel(0, random.randint(700, 1300))
103            self._human_delay(0.7, 1.8)
104
105        self._extract_visible_products(page)
106
107    def _extract_visible_products(self, page) -> None:
108        cards = self._first_locator_with_results(page,
               PRODUCT_CARD_SELECTORS)
109        if cards is None:
110            return
111
```

```
112         count = min(cards.count(), 120)
113         for index in range(count):
114             if len(self.products) >= MAX_PRODUCTS:
115                 return
116 
117             card = cards.nth(index)
118             product_url = self._extract_product_url(card)
119             if product_url in self._seen_urls:
120                 continue
121 
122             name = self._extract_text(card, PRODUCT_NAME_SELECTORS)
123             price = self._extract_price(card)
124 
125             if not name or not price:
126                 continue
127 
128             self._seen_urls.add(product_url)
129             self.products.append(
130                 {
131                     "nome": name,
132                     "prezzo": price,
133                     "url": product_url or "",
134                 }
135             )
136 
137     def _first_locator_with_results(self, page, selectors: List[str
        ]):
138         for selector in selectors:
139             locator = page.locator(selector)
140             try:
141                 if locator.count() > 0:
142                     return locator
143             except Exception:
144                 continue
145         return None
146 
147     def _extract_product_url(self, card) -> Optional[str]:
148         try:
149             link = card.locator(PRODUCT_LINK_SELECTOR).first
150             href = link.get_attribute("href", timeout=3000) if link
                .count() else card.get_attribute("href")
151             if not href:
152                 return None
153             if href.startswith("/"):
154                 return f"{BASE_URL}{href}"
155             return href
156         except Exception:
157             return None
158 
159     def _extract_text(self, card, selectors: List[str]) -> Optional
        [str]:
160         for selector in selectors:
161             try:
162                 locator = card.locator(selector).first
163                 if locator.count():
```

```
164                     text = self._clean_text(locator.inner_text(
                            timeout=3000))
165                     if text and CURRENCY not in text and len(text)
                            > 1:
166                         return text
167             except Exception:
168                 continue
169
170         try:
171             text = self._clean_text(card.inner_text(timeout=3000))
172             lines = [line for line in text.splitlines() if line and
                     CURRENCY not in line]
173             return lines[0] if lines else None
174         except Exception:
175             return None
176
177     def _extract_price(self, card) -> Optional[str]:
178         for selector in PRODUCT_PRICE_SELECTORS:
179             try:
180                 locator = card.locator(selector).first
181                 if locator.count():
182                     price = self._find_price(locator.inner_text(
                            timeout=3000))
183                     if price:
184                         return price
185             except Exception:
186                 continue
187
188         try:
189             return self._find_price(card.inner_text(timeout=3000))
190         except Exception:
191             return None
192
193     def _find_price(self, text: str) -> Optional[str]:
194         match = re.search(r"(\d+(?:[.,]\d{1,2})?)\s*EUR", text)
195         if not match:
196             return None
197         return match.group(1).replace(",", ".")
198
199     def _go_to_next_page(self, page) -> bool:
200         for selector in NEXT_PAGE_SELECTORS:
201             try:
202                 next_element = page.locator(selector).first
203                 if next_element.count() and next_element.is_visible
                        () and next_element.is_enabled():
204                     next_element.click()
205                     page.wait_for_load_state("domcontentloaded",
                            timeout=PAGE_LOAD_TIMEOUT_MS)
206                     self._human_delay(2.0, 4.5)
207                     return True
208             except Exception:
209                 continue
210         return False
211
212     def _clean_text(self, text: str) -> str:
```

```
213         return re.sub(r"\s+", " ", text).strip()
214
215     def _human_delay(self, minimum: float = MIN_DELAY_SECONDS,
            maximum: float = MAX_DELAY_SECONDS) -> None:
216         time.sleep(random.uniform(minimum, maximum))
217
218     def _save_json(self, result: Dict[str, List[Dict[str, str]]])
            -> None:
219         with open(OUTPUT_FILE, "w", encoding="utf-8") as file:
220             json.dump(result, file, ensure_ascii=False, indent=2)
221
222
223 if __name__ == "__main__":
224     data = Site_C().get_name_price()
225     print(json.dumps(data, ensure_ascii=False, indent=2))
```

Vediamo ora una breve panoramica del codice generato:

- `get_name_price(parametroRicerca)`: è il metodo entry point dello scraper; esso gestisce e configura le istanze di Playwright, effettua la chiamata all'url di ricerca e, infine, tutti restituisce i dati nel formato JSON;
- `_open_browser(self,playwright)`: è responsabile per l'inizializzazione della sessione; implementa un meccanismo di fallback se fallisce la prima prova;
- `_collect_products_from_page(self,page)`: gestisce il caricamento dinamico della pagina implementando un meccanismo di scroll per forzare il caricamento dei prodotti tramite il lazy-loading;
- `_extract_visible_products(self,page)` si occupa dell'estrazione dei dati dal DOM, estraendo nome e prezzo ed evitando i duplicati.

Di seguito vediamo in formato JSON il risultato dello scraping:


```
{
  "prodotti":[
    {
      "nome": "Tamaris",
      "prezzo": "5.00"
    },
    {
      "nome": "Bull Boys",
      "prezzo": "25.00"
    },
    {
      "nome": "Nike",
      "prezzo": "10.00"
    },
    {
      "nome": "Fred Perry",
      "prezzo": "20.00"
    },
    {
      "nome": "Crocs",
```

```
        "prezzo": "20.00"
    },
    {
        "nome": "New Balance",
        "prezzo": "25.00"
    },
    {
        "nome": "Unisa",
        "prezzo": "30.00"
    },
    {
        "nome": "Diadora",
        "prezzo": "8.00"
    },
    {
        "nome": "Frozen",
        "prezzo": "11.50"
    },
    {
        "nome": "Bugatti",
        "prezzo": "15.00"
    },
    {
        "nome": "Puma",
        "prezzo": "20.00"
    },
    {
        "nome": "Kiabi",
        "prezzo": "15.00"
    },
    {
        "nome": "Jade",
        "prezzo": "11.00"
    },
    {
        "nome": "Champion",
        "prezzo": "10.00"
    },
    {
        "nome": "Workout",
        "prezzo": "4.00"
    },
    {
        "nome": "Giorgio",
        "prezzo": "15.00"
    },
    {
        "nome": "Calvin Klein",
        "prezzo": "35.00"
    },
    {
        "nome": "Zara",
        "prezzo": "14.00"
    },
    {
```

```
        "nome": "Vera␣Pelle",
        "prezzo": "15.00"
      },
      {
        "nome": "Alviero␣Martini",
        "prezzo": "35.00"
      },
      {
        "nome": "&␣Other␣Stories",
        "prezzo": "10.00"
      },
      {
        "nome": "Birkenstock",
        "prezzo": "60.00"
      },
      {
        "nome": "Aerosoles",
        "prezzo": "18.00"
      },
      {
        "nome": "New␣Balance",
        "prezzo": "38.00"
      },
      {
        "nome": "New␣Balance",
        "prezzo": "57.80"
      },
      {
        "nome": "Bugatti",
        "prezzo": "15.00"
      },
      {
        "nome": "Laura␣Biagiotti",
        "prezzo": "12.00"
      },
      {
        "nome": "Twin␣Set␣by␣Simona␣Barbieri",
        "prezzo": "15.00"
      },
      {
        "nome": "Scott",
        "prezzo": "25.00"
      },
      {
        "nome": "Nike␣Air",
        "prezzo": "20.00"
      },
      {
        "nome": "Nike",
        "prezzo": "40.00"
      },
      {
        "nome": "Skechers",
        "prezzo": "26.00"
      },
```

```
{
  "nome": "Nike",
  "prezzo": "26.00"
},
{
  "nome": "adidas",
  "prezzo": "12.00"
},
{
  "nome": "Saucony",
  "prezzo": "65.00"
},
{
  "nome": "adidas",
  "prezzo": "70.00"
},
{
  "nome": "Nike",
  "prezzo": "50.00"
},
{
  "nome": "Peter␣Kaiser",
  "prezzo": "30.00"
},
{
  "nome": "Nike␣Air",
  "prezzo": "70.00"
},
{
  "nome": "GUESS",
  "prezzo": "25.00"
},
{
  "nome": "Dainese",
  "prezzo": "190.00"
},
{
  "nome": "Fashion",
  "prezzo": "15.00"
},
{
  "nome": "MNG",
  "prezzo": "30.00"
},
{
  "nome": "New␣Balance",
  "prezzo": "60.00"
},
{
  "nome": "Minelli",
  "prezzo": "25.00"
},
{
  "nome": "Munich",
  "prezzo": "16.00"
```

```
    },
    {
      "nome": "Birkenstock",
      "prezzo": "40.00"
    },
    {
      "nome": "Jordan",
      "prezzo": "40.00"
    },
    {
      "nome": "Nike",
      "prezzo": "10.00"
    },
    {
      "nome": "adidas",
      "prezzo": "50.00"
    }
  ]
}
```

5.4.3 Sviluppo dello scraper con il modello Claude Sonnet

Di seguito vediamo il codice generato da Claude Sonnet in risposta al prompt:

```python
class Site_C:

    def __init__(self):
        self._session = cffi_requests.Session(impersonate=
            TLS_IMPERSONATE)
        self._cookie: dict[str, str] = {}

    # -- cookie
        ---------------------------------------------------------------


    def _fetch_cookie(self, retries: int = 3) -> bool:
        """
        Visita la homepage per ottenere il cookie di sessione
        `access_token_web`, necessario per le chiamate API.
        """
        headers = {
            "Accept":                    "text/html,application/
                xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
            "Accept-Language":           ACCEPT_LANGUAGE,
            "Accept-Encoding":           "gzip, deflate, br",
            "Connection":                "keep-alive",
            "Upgrade-Insecure-Requests": "1",
            "DNT":                       "1",
            "Sec-Fetch-Dest":            "document",
            "Sec-Fetch-Mode":            "navigate",
            "Sec-Fetch-Site":            "none",
            "Sec-Fetch-User":            "?1",
        }
```

```
26
27          for attempt in range(retries):
28              try:
29                  print(f"  Recupero cookie (tentativo {attempt +
                        1}/{retries})...")
30                  resp = self._session.get(SITE_URL, headers=headers,
                         timeout=20)
31                  print(f"     Homepage status: {resp.status_code}")
32
33                  # Cerca il cookie nella risposta
34                  for name in [COOKIE_NAME, "access_token", "
                        _fr_session"]:
35                      val = resp.cookies.get(name)
36                      if val:
37                          self._cookie[name] = val
38                          print(f"     [OK] Cookie trovato: {name}={
                                val[:20]}...")
39                          return True
40
41                  if resp.status_code == 200:
42                      print("     [!] Cookie non trovato ma homepage
                            200 OK - si prova comunque")
43                      return True
44
45              except Exception as e:
46                  print(f"     [X] Errore homepage: {e}")
47
48              if attempt < retries - 1:
49                  sleep = 2 ** attempt + random.uniform(0, 1)
50                  print(f"     Attesa {sleep:.1f}s prima di riprovare
                        ...")
51                  time.sleep(sleep)
52
53          return False
54
55      # -- API call
            ---------------------------------------------------------------


56
57      def _api_search(self, page: int = 1) -> dict:
58          """Chiama l'endpoint interno /api/v2/catalog/items e
                restituisce il JSON."""
59          cookie_str = "; ".join(f"{k}={v}" for k, v in self._cookie.
                items())
60
61          headers = {
62              "Accept":           "application/json, text/plain, */*"
                    ,
63              "Accept-Language":  ACCEPT_LANGUAGE,
64              "Accept-Encoding":  "gzip, deflate, br",
65              "Connection":       "keep-alive",
66              "DNT":              "1",
67              "Referer":          REFERER,
68              "Origin":           SITE_URL,
69              "Sec-Fetch-Dest":   "empty",
```

```
70              "Sec-Fetch-Mode":   "cors",
71              "Sec-Fetch-Site":   "same-origin",
72          }
73          if cookie_str:
74              headers["Cookie"] = cookie_str
75
76          params = {
77              "search_text": SEARCH_QUERY,
78              "status_ids[]": CONDITION_ID,
79              "per_page":     str(PER_PAGE),
80              "page":         str(page),
81              "order":        "newest_first",
82          }
83
84          url = SITE_URL + API_SEARCH_PATH
85          print(f"  GET {url} (pagina {page})")
86
87          resp = self._session.get(url, headers=headers, params=
                params, timeout=25)
88          print(f"     Status: {resp.status_code} | Bytes: {len(resp.
                content)}")
89
90          if resp.status_code == 401:
91              print("     [!] 401 - rinnovo cookie...")
92              self._fetch_cookie()
93              # riprova una volta
94              resp = self._session.get(url, headers=headers, params=
                    params, timeout=25)
95              print(f"     Retry status: {resp.status_code}")
96
97          resp.raise_for_status()
98          return resp.json()
99
100     # -- parsing
            ---------------------------------------------------------------


101
102     @staticmethod
103     def _parse_price(raw) -> str:
104         """Normalizza il prezzo proveniente dall'API."""
105         if raw is None:
106             return "N/D"
107         # Formato dict: {"amount": "35.00", "currency_code": "EUR"}
108         if isinstance(raw, dict):
109             amount = raw.get("amount") or raw.get("amount_string")
                    or ""
110             return f"{CURRENCY_SYMBOL} {amount}".strip()
111         # Formato stringa o numero
112         return f"{CURRENCY_SYMBOL} {raw}".strip()
113
114     @staticmethod
115     def _parse_item(item: dict) -> dict:
116         """Estrae i campi rilevanti da un record item dell'API."""
117         price_raw = item.get("price")
118         price = Site_C._parse_price(price_raw)
```

```
119
120         # Taglia
121         size = (
122             item.get("size_title")
123             or item.get("size")
124             or "N/D"
125         )
126
127         # Brand
128         brand = (
129             item.get("brand_title")
130             or (item.get("brand_dto") or {}).get("title")
131             or "N/D"
132         )
133
134         # URL
135         url = item.get("url") or (SITE_URL + item.get("path", ""))
                or "N/D"
136
137         return {
138             "nome":       item.get("title", "N/D"),
139             "prezzo":     price,
140             "taglia":     size,
141             "marca":      brand,
142             "condizione": CONDITION_LABEL,
143             "url":        url,
144         }
145
146     # -- scraping core
            ------------------------------------------------------------
147
148     def _scrape(self) -> list[dict]:
149         """Scarica pagine successive fino a MAX_PRODUCTS prodotti.
                """
150         collected: list[dict] = []
151
152         # 1. Warm-up cookie
153         ok = self._fetch_cookie()
154         if not ok:
155             print("  [!] Impossibile ottenere il cookie - si tenta
                    comunque l'API")
156
157         # 2. Loop sulle pagine
158         for page in range(1, MAX_PAGES + 1):
159             if len(collected) >= MAX_PRODUCTS:
160                 break
161
162             print(f"\nPagina {page}/{MAX_PAGES}")
163
164             try:
165                 data = self._api_search(page)
166             except Exception as e:
167                 print(f"  [X] Errore API pagina {page}: {e}")
168                 break
169
```

```
170             items = data.get("items", [])
171             print(f"  [OK] Ricevuti {len(items)} item da API")
172 
173             if not items:
174                 print("  [i] Nessun risultato - fine paginazione")
175                 break
176 
177             for item in items:
178                 if len(collected) >= MAX_PRODUCTS:
179                     break
180                 collected.append(self._parse_item(item))
181 
182             # Controlla se ci sono ulteriori pagine
183             pagination = data.get("pagination", {})
184             total_pages = pagination.get("total_pages", 1)
185             print(f"  [i] Pagina {page}/{total_pages} - raccolti {
                    len(collected)}/{MAX_PRODUCTS}")
186 
187             if page >= total_pages:
188                 print("  [i] Ultima pagina raggiunta")
189                 break
190 
191             # Pausa anti-rate-limit
192             wait = WAIT_BETWEEN_PAGES + random.uniform(0,
                    WAIT_JITTER)
193             time.sleep(wait)
194 
195         return collected[:MAX_PRODUCTS]
196 
197     # -- API pubblica
            ------------------------------------------------------------
198 
199     def get_name_price(self) -> dict:
200         """
201         Esegue lo scraping e restituisce:
202             {
203                 "valuta": "EUR",
204                 "prodotti": [
205                     {"nome": "...", "prezzo": "EUR 35", "taglia":
                            "43", ...},
206                     ...
207                 ]
208             }
209         """
210         print(f"\nAvvio scraping: {SITE_URL}")
211         print(f"   Query:      {SEARCH_QUERY}")
212         print(f"   Condizione: {CONDITION_LABEL} (ID={CONDITION_ID
                    })")
213         print(f"   Target:     {MAX_PRODUCTS} prodotti\n")
214 
215         products = self._scrape()
216 
217         return {
218             "valuta":   CURRENCY_SYMBOL,
219             "prodotti": products,
```

```
        }


# ================================================================================
#  ENTRY POINT
# ================================================================================


if __name__ == "__main__":
    scraper = Site_C()

    try:
        data = scraper.get_name_price()
    except KeyboardInterrupt:
        print("\n[!] Interrotto dall'utente.")
        sys.exit(0)

    n = len(data["prodotti"])
    print(f"\n[OK] Prodotti recuperati: {n}")

    # Salva JSON
    output_file = "site_c_output.json"
    with open(output_file, "w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=2)
    print(f"Salvato in: {output_file}")

    # Anteprima console (primi 3)
    preview = {"valuta": data["valuta"], "prodotti": data["prodotti
        "][:3]}
    print("\nAnteprima (primi 3 prodotti):")
    print(json.dumps(preview, ensure_ascii=False, indent=2))

    if n == 0:
        print("\n[!] Nessun prodotto trovato.")
        print("   Possibili cause:")
        print("   1. Datadome ha bloccato la richiesta (prova a
            cambiare rete/VPN)")
        print("   2. I parametri di ricerca non hanno risultati con
             la condizione scelta")
        print("   3. Site C ha aggiornato le difese anti-bot")

    print(data)
```

Di seguito forniamo una breve panoramica del codice generato:

- `apply_search_and_filter(page)`: apre il sito, accetta i cookie, inserisce il nostro parametro di ricerca e applica la condizione richiesta.
- `with_page_param(url,page_number)`: modifica l'URL fornito aggiungendo o aggiornando il parametro "page"; così lo script è in grado di leggere i risultati nelle pagine successive.

- `clean_price(price_text)`: rimuove la valuta e lascia soltanto il valore numerico del prezzo.
- `get_model_price(price_text)`: avvia Playwright, apre un browser con profilo persistente, richiama ricerca o filtro e itera 50 prodotti su cui recupererà le informazioni. Successivamente, esegue un return fornendo il codice JSON nel formato richiesto nel prompt.

Di seguito vediamo in formato JSON il risultato dello scraping:


```
{
  "prodotti":[
    {
      "nome": "Scarpe",
      "prezzo": "16.0",
      "taglia": "39",
      "marca": "N/D",
      "condizione":"Nuovo senza cartellino"
    },
    {
      "nome": "New Balance 740 Sneakers - EU 38.5",
      "prezzo": "85.0",
      "taglia": "38.5",
      "marca": "New Balance",
      "condizione":"Nuovo senza cartellino"
    },
    {
      "nome": "Sneakers Puma bambini",
      "prezzo": "23.0",
      "taglia": "22",
      "marca": "Puma",
      "condizione":"Nuovo senza cartellino"
    },
    {
      "nome": "Zwarte ballerina's",
      "prezzo": "12.5",
      "taglia": "39",
      "marca": "H&M",
      "condizione":"Nuovo senza cartellino"
    },
    {
      "nome": "Zapato piel marron",
      "prezzo": "27.0",
      "taglia": "38",
      "marca": "Zara",
      "condizione":"Nuovo senza cartellino"
    },
    {
      "nome": "Sandalo infradito donna piu pochette colore nero",
      "prezzo": "6.0",
      "taglia": "40",
      "marca": "Donna",
      "condizione":"Nuovo senza cartellino"
    },
    {
```

```
    "nome": "Basket neuf T.25 8e",
    "prezzo": "8.0",
    "taglia": "25",
    "marca": "Orchestra",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Sandalias de piel serraje Mango",
    "prezzo": "25.0",
    "taglia": "37",
    "marca": "Mango",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Bambas de Nike",
    "prezzo": "40.0",
    "taglia": "45",
    "marca": "Nike",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Basket blanche Adidas response cushion 16 neuve
      homme",
    "prezzo": "35.0",
    "taglia": "42",
    "marca": "adidas",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Witte opengewerkte schoenen AGL maat 38",
    "prezzo": "30.0",
    "taglia": "38",
    "marca": "AGL",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Mooie nieuwe veterschoenen van Waldlaufer",
    "prezzo": "59.0",
    "taglia": "40",
    "marca": "Waldlaufer",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Adidas Originals Adilette Badeschlappen",
    "prezzo": "36.99",
    "taglia": "37",
    "marca": "adidas",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "MT5 TOPO Taille 42",
    "prezzo": "115.0",
    "taglia": "42",
    "marca": "Topo Athletic",
    "condizione": "Nuovo senza cartellino"
```

```
    },
    {
      "nome": "Air␣Max␣DN␣pointure␣33,5",
      "prezzo": "45.0",
      "taglia": "33",
      "marca": "Nike",
      "condizione": "Nuovo␣senza␣cartellino"
    },
    {
      "nome": "Sandale␣plates␣Calank␣cuir␣T40␣-␣Neuves",
      "prezzo": "24.65",
      "taglia": "40",
      "marca": "Calank",
      "condizione": "Nuovo␣senza␣cartellino"
    },
    {
      "nome": "CR7␣crampon␣Njke␣Mercurial␣Superfly␣RGN␣SE",
      "prezzo": "385.0",
      "taglia": "44",
      "marca": "Nike",
      "condizione": "Nuovo␣senza␣cartellino"
    },
    {
      "nome": "Superga␣Basket␣montante␣38␣White/blanc␣Alpina",
      "prezzo": "20.0",
      "taglia": "38",
      "marca": "Superga",
      "condizione": "Nuovo␣senza␣cartellino"
    },
    {
      "nome": "Zapatos␣de␣salon␣nude/beige␣de␣ante␣-␣ta con␣ancho␣
          comodo",
      "prezzo": "10.0",
      "taglia": "38",
      "marca": "H&M",
      "condizione": "Nuovo␣senza␣cartellino"
    },
    {
      "nome": "Chaussure␣femme",
      "prezzo": "7.0",
      "taglia": "38",
      "marca": "Marina",
      "condizione": "Nuovo␣senza␣cartellino"
    },
    {
      "nome": "Chaussures,␣cuir",
      "prezzo": "14.0",
      "taglia": "41",
      "marca": "N/D",
      "condizione": "Nuovo␣senza␣cartellino"
    },
    {
      "nome": "Chaussures␣trail␣361␣Taroko␣5",
      "prezzo": "75.0",
      "taglia": "42",
```

```
    "marca": "361 Degrees",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Chinelo homem",
    "prezzo": "45.0",
    "taglia": "46",
    "marca": "Reef",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Chanclas",
    "prezzo": "2.0",
    "taglia": "17",
    "marca": "Primark",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "The Nike premier III FG talla 40,5",
    "prezzo": "59.0",
    "taglia": "40,5",
    "marca": "Nike",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Paire de on cloud",
    "prezzo": "70.0",
    "taglia": "42",
    "marca": "On Cloud",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Crampons",
    "prezzo": "60.0",
    "taglia": "41",
    "marca": "Nike",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "NEU Ozweego Kids Grosse 21",
    "prezzo": "25.0",
    "taglia": "21",
    "marca": "adidas",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Scarpe uomo hey Dude numero 44",
    "prezzo": "55.0",
    "taglia": "44",
    "marca": "Hey Dude",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Nike Sneaker",
    "prezzo": "60.0",
```

```
    "taglia": "36.5",
    "marca": "Nike",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Balanciagx Adidas",
    "prezzo": "500.0",
    "taglia": "44",
    "marca": "Balenciaga",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Basket asics neuves",
    "prezzo": "57.0",
    "taglia": "43,5",
    "marca": "Asics",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Crampons Nike JR phantom gx 2 club",
    "prezzo": "28.0",
    "taglia": "36",
    "marca": "Nike",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Basket femme Hugo pointure 39 authentique",
    "prezzo": "135.0",
    "taglia": "39",
    "marca": "Hugo Boss",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "jordan 3 jbalvin beige certificate POIZON",
    "prezzo": "320.0",
    "taglia": "43",
    "marca": "Jordan",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Nike Sabrina 3",
    "prezzo": "129.99",
    "taglia": "42",
    "marca": "Nike",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Claquette en liege mixte",
    "prezzo": "18.0",
    "taglia": "42",
    "marca": "Gumbies",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Basket Manoukian Paris beige - Taille 41",
```

```
    "prezzo": "50.0",
    "taglia": "41",
    "marca": "Manoukian",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Infradito bimba n.33 (B)",
    "prezzo": "5.0",
    "taglia": "33",
    "marca": "Champion",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Chinelo Havaianas - Novos",
    "prezzo": "17.0",
    "taglia": "43",
    "marca": "Havaianas",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Mules negro piel vegana sin estrena rpuntera
        cuadrada",
    "prezzo": "6.0",
    "taglia": "39",
    "marca": "sin marca",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "sandalo donna in vera pelle nera n36",
    "prezzo": "39.9",
    "taglia": "36",
    "marca": "Lorenzo Mari",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Nike P-6000 T43",
    "prezzo": "105.0",
    "taglia": "43",
    "marca": "Nike",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Crocs",
    "prezzo": "28.0",
    "taglia": "36",
    "marca": "Crocs",
    "condizione": "Nuovo senza cartellino"
  },
  {
    "nome": "Claquette ete",
    "prezzo": "10.0",
    "taglia": "38",
    "marca": "N/D",
    "condizione": "Nuovo senza cartellino"
  },
```

```
      {
        "nome": "Groundies␣Hamptons␣2.0",
        "prezzo": "45.0",
        "taglia": "44",
        "marca": "GROUNDIES",
        "condizione": "Nuovo␣senza␣cartellino"
      },
      {
        "nome": "Basket␣avengers␣taille␣37␣GEOX",
        "prezzo": "29.0",
        "taglia": "37",
        "marca": "Geox",
        "condizione": "Nuovo␣senza␣cartellino"
      },
      {
        "nome": "Botas␣de␣piel",
        "prezzo": "70.0",
        "taglia": "39",
        "marca": "Biu␣cili",
        "condizione": "Nuovo␣senza␣cartellino"
      },
      {
        "nome": "asics␣gel␣nyc",
        "prezzo": "80.0",
        "taglia": "42",
        "marca": "Asics",
        "condizione": "Nuovo␣senza␣cartellino"
      },
      {
        "nome": "Schuhe",
        "prezzo": "33.0",
        "taglia": "38",
        "marca": "adidas",
        "condizione": "Nuovo␣senza␣cartellino"
      }
    ]
  }
```


5.4.4 Valutazione comparativa sul sito target

Nella Tabella 5.4 mettiamo a confronto i modelli di Intelligenza Artificiale utilizzati con i requisiti funzionali e non funzionali.

5.5 Discussione e linee guida pratiche

L'analisi dei risultati ricavati dal processo di sperimentazione ha rilevato differenze consistenti tra Claude Sonnet e OpenAI Codex.

Le differenze principali sulle generazioni di codice possono così essere riassunte:

- *efficienza del codice*: Codex ha mostrato una capacità migliore utilizzando lo stesso prompt, fornendo codice più pulito e funzionante;
- *spiegazione del codice*: Sonnet, rispetto all'altro modello, ha generato molti più commenti esplicativi, come richiesto;

Requisiti	**OpenAI Codex**	**Claude Sonnet**
RF1 - Generazione del codice in Python	Soddisfatto	Soddisfatto
RF2 - Utilizzo della libreria Playwright	Soddisfatto	Soddisfatto
RF3 - Conformità dei dati in JSON	Soddisfatto	Non Soddisfatto
RF4 - Gestione della paginazione	Soddisfatto	Soddisfatto
RF5 - Eliminazione della valuta	Soddisfatto	Soddisfatto
RF6 - Codice senza stampa	Soddisfatto	Non Soddisfatto
RF7 - Gestione del caricamento	Soddisfatto	Soddisfatto
RF8 - Intervento umano assente	Soddisfatto	Non Soddisfatto
RNF1 - Codice commentato	Non Soddisfatto	Soddisfatto
RNF2 - Selettore minimo	Soddisfatto	Soddisfatto

Tabella 5.4: Confronto dei requisti tra i modelli utilizzati

- *rispetto dei vincoli*: entrambi i modelli presentano un'aderenza scarsa al prompt e mostrano, in più occasioni, l'utilizzo del metodo `print()` esplicitamente escluso;
- *difficoltà di elaborazione*: Sonnet ha mostrato problematiche durante la generazione del codice, visto che non riusciva ad accedere all'HTML del sito di interesse.

Per ottimizzare lo sviluppo del codice per lo scraping abbiamo individuato delle linee guida. Vediamo le più rilevanti:

- *HTML anticipato*: per migliorare la generazione del codice per lo scraping, abbiamo notato che, se al modello vengono forniti gli elementi direttamente nel prompt, esso genera un risultato migliore;
- *attesa intelligente*: nel momento in cui il sistema accede ad un determinato sito web, spesso risulta opportuno far attendere l'algoritmo per far caricare la pagina di interesse; in questo modo, non si rilevano errori durante l'estrazione degli elementi;
- *gestione delle eccezioni*: entrambi i modelli scarseggiano di meccanismi protettivi per la gestione degli errori; questo problema potrebbe essere risolto inserendo all'interno del prompt il vincolo sulla sicurezza.

La sperimentazione condotta sulla generazione di codice con i sistemi di Intelligenza Artificiale ha rilevato che Codex OpenAI ha ottenuto risultati migliori negli aspetti di maggiore interesse, cioè l'estrazione di dati da siti web. Sonnet, invece, si è dimostrato utile soprattutto nella parte della spiegazione del codice generato, mostrando all'utente numerosi commenti.

Nonostante le difficoltà iniziali rilevate nell'analisi del codice HTML e l'estrazione di liste vuote dovute a cause secondarie, siamo riusciti ad estrarre numerosi dati con tempi minori rispetto alla programmazione manuale dello stesso meccanismo. L'utilizzo dell'Intelligenza Artificiale nella generazione di codice è stato utile per prototipare velocemente i meccanismi di scraping.

CAPITOLO 6

Discussione

In questo capitolo viene presentata la discussione dei case study analizzati e sviluppati nelle sezioni precedenti.

Divideremo la discussione dei risultati in parti distinte, ciascuna dedicata a uno specifico case study.

6.1 Introduzione alla discussione

L'obiettivo del nostro studio è stato quello di testare i sistemi di Intelligenza Artificiale Generativa sui dei problemi reali che possono presentarsi in diversi contesti.

Nei capitoli precedenti sono stati trattati tre differenti case study riguardanti il testo, le immagini e il codice. I risultati sviluppati hanno evidenziato peculiarità positive, criticità e metodologie di miglioramento. In questo capitolo proporremo una panoramica generale, spostando il focus sulla visione globale dei risultati ottenuti ed effettuando dei parallelismi tra le diverse tecnologie utilizzate, con l'obiettivo di individuare gli strumenti che si sono rivelati più utili nel nostro compito.

6.2 Discussione sul case study per il testo

In questo case study abbiamo analizzato un problema realmente gestito con due modelli di Intelligenza Artificiale, ovvero ChatGPT e Gemini.

Entrambi i modelli hanno avuto input iniziali identici ma sviluppi differenti, dimostrando un'ottima conoscenza della problematica da noi analizzata. Risulta, però, evidente che, di fronte a scelte di fattibilità economica e di tempo necessario, allo svolgimento, essi hanno suggerito soluzioni che non sono risultate pienamente soddisfacenti a causa dei vincoli stringenti che avevamo imposto.

6.3 Discussione sul case study per le immagini

Lo sviluppo di questo case study è stato attuato mediante l'utilizzo di ChatGPT e Gemini.

Durante l'approccio iniziale, in cui è stato presentato il prompt di partenza, è stata rilevata una difficoltà, da parte di entrambi i modelli di Intelligenza Artificiale, nella comprensione e nell'esecuzione dei vincoli stabiliti.

I modelli di Intelligenza Artificiale tendevano a rimanere bloccati in un ciclo continuo in cui presentavano la stessa soluzione nonostante i numerosi tentativi, la rigenerazione della sessione e la riprogettazione del prompt. I risultati ottenuti derivano da alcune prove effettuate variando il prompt iniziale. Nel momento in cui il problema iniziale è stato superato, gli strumenti utilizzati hanno mostrato un'ottima comprensione ed elaborazione delle interfacce che sono state richieste.

6.4 Discussione sul case study per il codice

Il case study per il confronto tra Codex e Claude Sonnet ha evidenziato profonde differenze tra le soluzioni presentate dai due modelli.

Dopo l'introduzione dei requisiti funzionali e non funzionali, sono stati sottoposti ai due modelli tre differenti siti web ai fini della sperimentazione. I risultati ottenuti mostrano un netto vantaggio di Codex rispetto a Claude Sonnet; quest'ultimo, nonostante l'esecuzione di numerose prove, non disponeva di strumenti necessari per affrontare il problema posto. La difficoltà più rilevante è stata il recupero del codice HTML, da parte del modello, direttamente dalla piattaforma web. Vista la posizione dichiarata di Anthropic in merito alla sicurezza della propria tecnologia, il modello potrebbe non essere riuscito ad accedere al sito web proprio per tali motivi. Codex, d'altro canto, si è dimostrato più diretto alla soluzione del problema, presentando proposte funzionali alla prima prova.

Conclusioni

Nella presente tesi è stato affrontato il tema dell'introduzione dell'Intelligenza Artificiale Generativa per la risoluzione di problemi articolati sotto forma di case study. Inizialmente è stata introdotta l'AI dal punto di vista teorico, mediante un'analisi dei meccanismi di funzionamento di questa tecnologia. Successivamente è stata trattata storia che ha portato all'attuale avanzamento tecnologico. Nei capitoli successivi abbiamo messo alla prova, attraverso dei case study progettati appositamente, i seguenti sistemi di Intelligenza Artificiale: Gemini, ChatGPT, Codex e Claude.

L'obiettivo della sperimentazione è stato quello di analizzare il comportamento dei modelli di fronte a casi d'uso appartenenti a diversi contesti. Ciò ci ha condotto alla sperimentazione con tre differenti case study che, vista la loro natura, hanno saputo mettere in evidenza differenti aspetti dell'attività di testing.

Il case study sul testo ci ha permesso di capire come un LLM possa valutare una situazione reale descritta a parole e aiutarci nel raggiungere la soluzione al problema posto. Il case study delle immagini ci ha permesso di valutare le capacità e i limiti della generazione delle immagini. Il case study sul codice ci ha permesso di osservare come il codice generato sotto un profilo funzionale e reale.

Osservando i risultati ottenuti abbiamo potuto trarre le nostre conclusioni prendendo coscienza sia dei punti di forza che di quelli di debolezza. Abbiamo, inoltre, potuto migliorare le nostre skill nella progettazione dei prompt da fornire agli LLM.

Una nota doverosa che dobbiamo mettere in luce sta nel fatto che, in quasi la totalità dei case study analizzati, l'intervento umano è stato di vitale importanza, sia per l'individuazione del problema che per la sua correzione tramite dei prompt secondari. Inoltre, i risultati ottenuti rispecchiano fedelmente le informazioni di utilizzo che abbiamo degli LLM al momento prodotti dalla comunità dei utilizzatori. Gli LLM comunemente utilizzati, nonostante siano degli strumenti utili, possiedono ancora delle criticità di comprensione visto che vengono definiti come "ottimi indovinatori".

Si evidenzia un possibile sviluppo futuro per alcuni dei case study presi in esame, in particolare il case study che riguarda la generazione delle interfacce utente per le ambulanze. Dopo lo sviluppo delle interfacce mediante l'AI, si potrebbe richiedere una generazione di codice sorgente in qualsiasi linguaggio per valutare se questa tecnologia è in grado di effettuare una sorta di *reverse engineering*, visto che partiamo da un'immagine e non dall'interfaccia stessa per la sua produzione. In tale scenario, il designer si limiterebbe ad effettuare degli aggiustamenti finali sull'interfaccia utente prodotta.

Per quanto riguarda il case study sul codice, si potrebbe sperimentare un approccio basato sulla programmazione mediante l'Intelligenza Artificiale che generi nuovo codice ogni qual

volta un sito web cambia il proprio set di parametri interni, rendendo, così, l'algoritmo adattabile automaticamente nel tempo alle variazioni del DOM del sito web.

Bibliografia

BROWN , T., MANN , B., RYDER , N., SUBBIAH , M., KAPLAN , J. D., DHARIWAL , P., NEELAKANTAN , A., SHYAM , P., SASTRY , G., ASKELL , A. e OTHERS (2020), «Language models are few-shot learners», *Advances in neural information processing systems*, vol. 33, p. 1877–1901.

DÍAZ-RODRÍGUEZ , N., DEL SER , J., COECKELBERGH , M., DE PRADO , M. L., HERRERA-VIEDMA , E. e HERRERA , F. (2023), «Connecting the dots in trustworthy Artificial Intelligence: From AI principles, ethics, and key requirements to responsible AI systems and regulation», *Information Fusion*, vol. 99, p. 101 896.

DOHMATOB , E., FENG , Y., SUBRAMONIAN , A. e KEMPE , J. (2025), «Strong model collapse», in «International Conference on Learning Representations», vol. 2025, p. 15 656–15 691.

DUHIGG , C. (2012), *The power of habit: Why we do what we do in life and business*, vol. 34, Random House.

GLEZ-PEÑA , D., LOURENÇO , A., LÓPEZ-FERNÁNDEZ , H., REBOIRO-JATO , M. e FDEZ-RIVEROLA , F. (2014), «Web scraping technologies in an API world», *Briefings in bioinformatics*, vol. 15 (5), p. 788–797.

JÄRVELÄ , S., NGUYEN , A. e HADWIN , A. (2023), «Human and artificial intelligence collaboration for socially shared regulation in learning», *British Journal of Educational Technology*, vol. 54 (5), p. 1057–1076.

MITCHELL , R. (2015), *Web scraping with Python: collecting data from the modern web*, " O'Reilly Media, Inc.".

TOUVRON , H., LAVRIL , T., IZACARD , G., MARTINET , X., LACHAUX , M.-A., LACROIX , T., ROZIÈRE , B., GOYAL , N., HAMBRO , E., AZHAR , F. e OTHERS (2023), «Llama: Open and efficient foundation language models», *arXiv preprint arXiv:2302.13971*.

TSAI , Y.-H. H., BAI , S., LIANG , P. P., KOLTER , J. Z., MORENCY , L.-P. e SALAKHUTDINOV , R. (2019), «Multimodal transformer for unaligned multimodal language sequences», in «Proceedings of the 57th annual meeting of the association for computational linguistics», p. 6558–6569.

TURING , A. M. (1980), «Computing machinery and intelligence.», *Creative Computing*, vol. 6 (1), p. 44–53.

TURING , A. M. (2007), «Computing machinery and intelligence», in «Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer», p. 23–65, Springer Netherlands Dordrecht.

VASWANI , A., S HAZEER , N., P ARMAR , N., U SZKOREIT , J., J ONES , L., G OMEZ , A. N., K AISER , Ł. e P OLOSUKHIN , I. (2017), «Attention is all you need», *Advances in neural information processing systems*, vol. 30.

WU, X.-K., C HEN , M., L I, W., WANG , R., L U, L., L IU , J., H WANG , K., H AO , Y., PAN , Y., MENG , Q. e OTHERS (2025), «Llm fine-tuning: Concepts, opportunities, and challenges», *Big Data and Cognitive Computing*, vol. 9 (4), p. 87.

YAO , Y., DUAN , J., X U, K., C AI , Y., SUN , Z. e Z HANG , Y. (2024), «A survey on large language model (llm) security and privacy: The good, the bad, and the ugly», *High-Confidence Computing*, vol. 4 (2), p. 100 211.

ZHANG , R., H AN , J., L IU , C., G AO , P., ZHOU , A., H U, X., Y AN , S., L U, P., LI, H. e Q IAO , Y. (2023), «Llama-adapter: Efficient fine-tuning of language models with zero-init attention», *arXiv preprint arXiv:2303.16199*.

Sitografia

- Anthropic - https://www.anthropic.com
- Anthropic Claude - https://platform.claude.com
- Apple Support - https://support.apple.com
- Claude Support - https://support.claude.com
- EU Artificial Intelligence Act - https://www.artificialintelligenceact.eu/the-act/
- Eur Lex - https://eur-lex.europa.eu
- Google - https://gemini.google
- Google Cloud - https://cloud.google.com
- Google Gemini - https://gemini.google.com
- Google Gemini API - https://ai.google.dev
- Google Trends - https://trends.google.it
- News from Google - https://blog.google
- OpenAI - https://www.openai.com
- OpenAI ChatGPT - https://chatgpt.com
- OpenAI Codex - https://openai.com/index/codex/
- Oxford English Dictionary - https://www.oed.com
- Webscraper - https://www.webscraper.io
- Wikipedia - https://it.wikipedia.org

Ringraziamenti

Un particolare ringraziamento va al Prof. Domenico Ursino che, grazie alla sua disponibilità, professionalità e guida autorevole, ha reso possibile la stesura di questa tesi, con la quale si conclude un mio importante capitolo personale.

Un profondo ringraziamento va a tutte le persone che mi sono state accanto durante questo percorso universitario: ciascuno di voi sa quanto sia stato prezioso il proprio contributo e io so bene quanto mi abbiate sostenuto.

Grazie a tutti voi.