



Università
di Catania

DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA TRIENNALE IN INFORMATICA

Lorenzo Maria Alberto Paoria

Confronto fra risposte di StackOverflow e ChatGpt per domande
riguardanti lo sviluppo di software

RELAZIONE PROGETTO FINALE

Relatore: Prof. Emiliano Tramontana

Correlatori:
Prof. Gabriella Verga
Dott. Alessandro Midolo

Anno Accademico 2023 – 2024

Indice

Capitolo 1 - Introduzione	4
Capitolo 2 - Metodologia	7
2.1 - Dataset StackOverlfow.....	8
2.2 - Estrazione e catalogazione dei post.....	9
2.2.1 - Estrazione.....	10
2.2.2 - Catalogazione.....	11
2.3 - Elaborazione dei post per catalogazione	13
2.4 - Analisi dei risultati.....	17
Capitolo 3 - Risultati	20
3.1 - Post con codice	20
3.1.1 - Esempi post con codice.....	21
3.2 - Post con caratteri superiori e inferiori a 700	28
3.2.1 - Esempi post con caratteri inferiori e superiori a 700.	29
3.3 - Catalogazioni per TF-IDF	38
3.3.1 - Esempi di post classificati secondo il termine "Programming"	38
3.4 - Considerazioni sui risultati	39
3.4.1 - Considerazioni risultati dei post con codice	39
3.4.2 - Considerazioni risultati dei post con caratteri inferiori e superiori a 700	40
3.4.3 - Considerazioni per la catalogazione secondo TF-IDF.....	41
3.4.4 - Considerazioni generali.....	41
Capitolo 4 - Conclusioni.....	43
Bibliografia	45

Capitolo 1 - Introduzione

Questa tesi si propone di confrontare le risposte fornite dagli utenti di StackOverflow con quelle generate da sistemi avanzati di intelligenza artificiale, come ChatGpt, per valutare l'efficacia di queste tecnologie nel fornire soluzioni tecniche di alta qualità per i programmatori. L'importanza di questo argomento è strettamente legata all'attuale diffusione e utilizzo crescente dell'intelligenza artificiale nel supporto allo sviluppo software. Con il continuo progresso delle tecnologie di elaborazione del linguaggio naturale, diventa essenziale capire se e in che misura tali strumenti possano competere con il contributo umano, dimostrando quindi il loro potenziale nel migliorare la produttività e la qualità del lavoro dei programmatori.

L'obiettivo principale è quindi quello di analizzare la pertinenza, l'accuratezza e l'affidabilità delle risposte prodotte dall'intelligenza artificiale rispetto a quelle offerte da esperti umani, sia per le domande prettamente testuali che per le domande contenenti codice, al fine di comprendere i limiti e le potenzialità delle moderne tecnologie di elaborazione del linguaggio naturale nel supportare la risoluzione di problemi complessi nel campo della programmazione, contribuendo così al miglioramento del lavoro dei programmatori a livello globale.

Per questo scopo le principali tecnologie che verranno usate sono:

- **StackOverflow** [1], è uno dei siti web più utilizzati dalla community di sviluppatori per porre domande su una vasta gamma di argomenti legati alla programmazione, e non solo. Gli utenti possono pubblicare domande su problematiche tecniche, algoritmi, linguaggi di programmazione, strumenti di sviluppo e altre tematiche. Una volta che una domanda viene pubblicata, diventa visibile a tutta la community, che ha la possibilità di fornire risposte. Le risposte possono essere votate dagli altri utenti: le

risposte con il maggior numero di voti vengono messe in evidenza come "migliori risposte", in quanto ritenute dalla community le più corrette o soddisfacenti. Questo meccanismo consente non solo di valorizzare le risposte più pertinenti, ma anche di offrire a chi consulta il sito una soluzione rapida e affidabile al proprio problema.

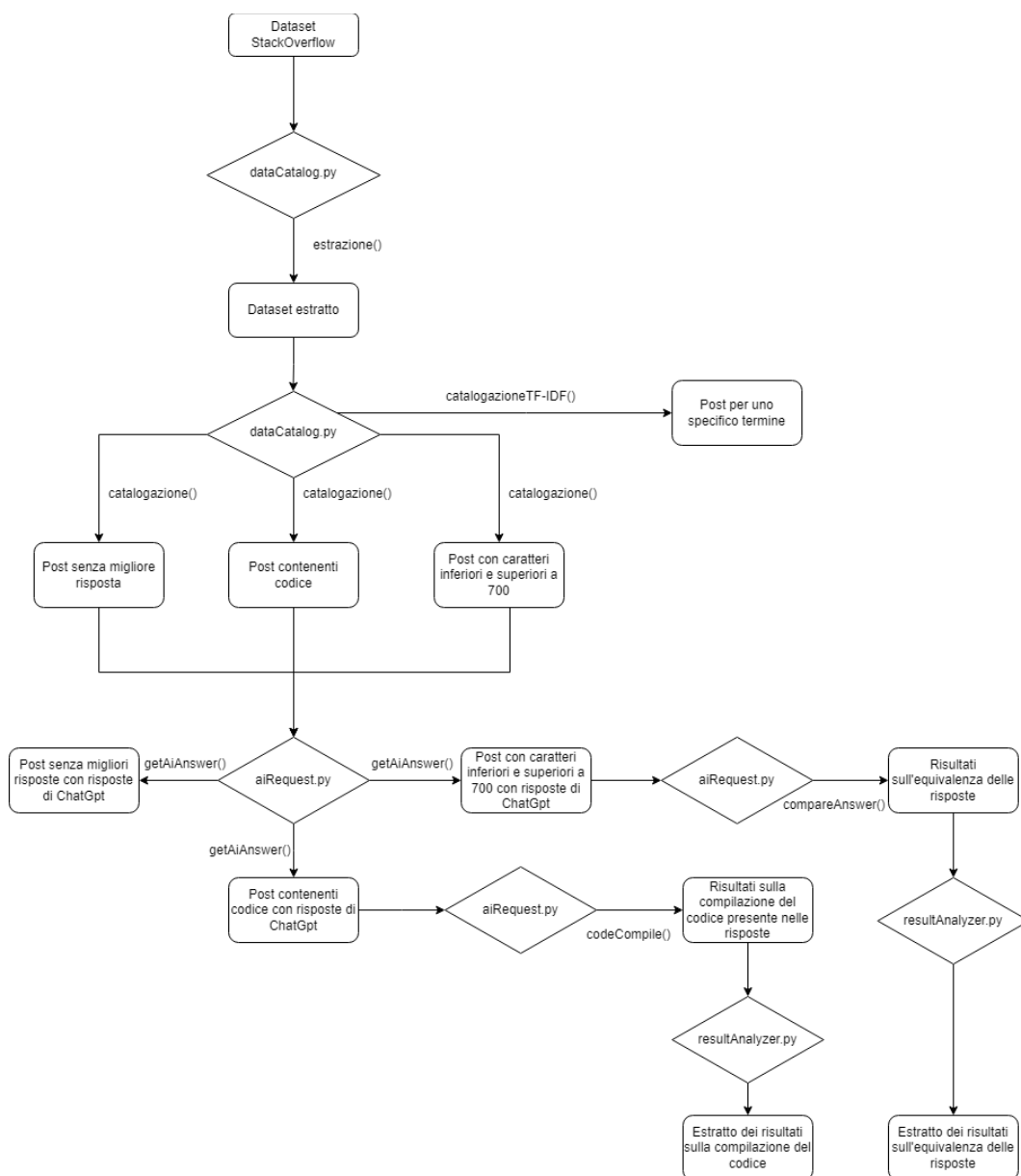
- **Sistemi di Intelligenza Artificiale (IA)**, Progettati per simulare l'intelligenza umana, questi sistemi sfruttano ampie raccolte di dati, seguite da una fase di progettazione e programmazione di modelli predittivi. I modelli di intelligenza artificiale, come quelli utilizzati per il riconoscimento vocale, la visione artificiale e la generazione di linguaggio naturale, si basano su sofisticati algoritmi che analizzano grandi quantità di informazioni per apprendere modelli e relazioni utili a compiere previsioni o rispondere a domande. Nel nostro caso, utilizzeremo un modello di generazione di testo per rispondere alle domande in modo automatico e contestualmente rilevante. Nello specifico, useremo ChatGpt [2], una chatbot sviluppata da OpenAI. Questo sistema è basato su modelli di linguaggio avanzati, addestrati su un vasto corpo di dati testuali provenienti da una varietà di fonti, inclusi libri, articoli, siti web, e altro ancora. Grazie a questo processo di addestramento, ChatGpt è in grado di comprendere e generare risposte coerenti, pertinenti e adattate al contesto delle domande che gli vengono poste. In particolare, ChatGpt si distingue per la sua capacità di interpretare richieste complesse, fornire spiegazioni dettagliate e rispondere con un linguaggio naturale, facilitando così l'interazione tra esseri umani e macchine. È progettato per assistere in molteplici scenari, tra cui la risoluzione di problemi tecnici, la fornitura di informazioni, o semplicemente l'interazione sociale, rappresentando un notevole progresso nella tecnologia delle chatbot e dei sistemi di intelligenza artificiale.

Nel seguito, verranno illustrati i contenuti dei capitoli della tesi.

- **Metodologia:** La sezione dedicata alla metodologia si concentrerà sull'approccio adottato per condurre l'analisi, fornendo una descrizione dettagliata di tutte le tecniche e gli strumenti utilizzati. Saranno esaminati i criteri di selezione dei dati e delle fonti, spiegando il processo di raccolta delle informazioni da StackOverflow e di catalogazione di essi. Inoltre verranno esposti i metodi di richiesta all'IA per la raccolta delle risposte e relative valutazioni su di esse tramite richieste API. Oltre ad una fase di raccolta dei dati finali per estrapolarne i risultati.
- **Risultati:** Questa sezione presenterà i risultati ottenuti dall'analisi condotta, fornendo una panoramica completa e dettagliata delle osservazioni emerse. Saranno presentati esempi pratici delle risposte provenienti sia da StackOverflow che da ChatGpt, evidenziando le somiglianze e le differenze tra le due. Verranno spiegate anche le situazioni in cui una delle due tecnologie ha dimostrato di essere più efficiente, precisa o pertinente rispetto all'altra. I risultati non solo metteranno in evidenza l'efficacia di entrambe le tecnologie nel risolvere problemi tecnici, ma ne mostreranno anche i limiti, fornendo una base concreta per ulteriori interpretazioni e discussioni.
- **Conclusione:** Nella conclusione, i risultati ottenuti saranno attentamente interpretati e analizzati, allo scopo di estrapolarne il significato più profondo. Verrà fornita una sintesi delle principali implicazioni emerse dall'analisi, mettendo in evidenza come le moderne tecnologie di intelligenza artificiale, come ChatGpt, possano essere utilizzate in modo complementare rispetto alle soluzioni offerte da piattaforme collaborative come StackOverflow. Si discuterà inoltre delle prospettive future, suggerendo possibili miglioramenti delle tecnologie esaminate e le potenziali aree di ricerca futura.

Capitolo 2 - Metodologia

In questo capitolo verranno descritti in dettaglio i passaggi e le procedure adottate per ottenere i risultati della presente ricerca. Questo capitolo ha l'obiettivo di fornire una chiara e trasparente panoramica del processo metodologico seguito, al fine di garantire la riproducibilità dello studio e la comprensione dei dati raccolti.



Il diagramma di flusso riportato spiega il funzionamento generale dei programmi, i dettagli verranno spiegati nelle sezioni successive.

2.1 - Dataset StackOverflow

Per prima cosa, e sicuramente la più importante, ho acquisito un dataset contenente i post del sito StackOveflow[1], una delle principali piattaforme online dove sviluppatori di tutto il mondo imparano, condividono conoscenze e costruiscono le proprie carriere. Per ottenere questo dataset, ho utilizzato il sito Archive.org [3], che mette a disposizione gratuitamente una copia completa di tutti i post pubblicati su StackOverflow.

Il dataset si presenta sotto forma di un file XML, contenente sia tutte le domande che tutte le risposte correlate. Ogni post include dettagli rilevanti come l'identificativo unico della domanda, l'utente che ha pubblicato la domanda, il testo della domanda stessa e una serie di risposte fornite dagli utenti. Ogni risposta di ogni post contiene un identificativo, il testo della risposta, l'ID dell'autore, la data di pubblicazione e un punteggio associato, che rappresenta il numero di voti positivi ricevuti dagli utenti della piattaforma.

Il dataset non si limita a includere solo le domande e le risposte, ma comprende anche altre informazioni archiviate per ogni sito della piattaforma: Posts, Users, Votes, Comments, Badges, Tags, PostHistory, e PostLinks.

Il dataset, disponibile a partire dal Gennaio 2014 sul sito Internet Archive, contiene numerose sezioni relative a diversi argomenti. Tuttavia, per la nostra analisi, ci concentriamo esclusivamente su una delle sezioni del dataset, quella relativa a Software Engineering. Da questa specifica parte, che contiene 1.283 post, prendiamo in considerazione solo le domande e le relative risposte, focalizzandoci

particolarmente sulle risposte migliori, ovvero quelle con il punteggio più alto, per garantire un'analisi precisa e mirata sulla qualità delle risposte. Nel caso in cui una domanda abbia delle risposte che non hanno ricevuto voti, queste vengono considerate come domande senza risposta.

2.2 - Estrazione e catalogazione dei post

Una volta ottenuto il dataset, è stato creato il programma denominato dataCatalog. L'obiettivo principale di questo programma è estrarre i post dal file XML e organizzarli in categorie specifiche in base a determinati criteri. Ovvero per i post contenenti codice e i post con un numero superiore e inferiore a 700 caratteri. Successivamente, i dati vengono strutturati e salvati in formato JSON per facilitare ulteriori analisi e utilizzi.

In questo programma sono state utilizzate diverse librerie Python, ciascuna con uno scopo specifico, per gestire ed elaborare i dati estratti dal file XML. Ecco una panoramica delle librerie impiegate e delle loro funzionalità principali:

- **lxml** [4]: Una libreria potente e veloce per il parsing di documenti XML e HTML. In questo programma, lxml è stata utilizzata per analizzare il file XML di StackOverflow e estrarre le informazioni rilevanti dai post, come il testo delle domande e delle risposte.
- **BeautifulSoup (bs4)** [5]: Utilizzata per la pulizia e l'elaborazione del contenuto HTML presente nei post. BeautifulSoup consente di rimuovere il markup HTML dai testi delle domande e delle risposte, lasciando solo il contenuto testuale necessario per l'analisi.
- **sklearn.feature_extraction** [6]: Questa sottolibreria di scikit-learn include strumenti per l'estrazione di caratteristiche da testi. In particolare, il programma utilizza TfidfVectorizer per creare una matrice TF-IDF, che

permette di quantificare l'importanza relativa di termini specifici all'interno dei post.

- **nltk (Natural Language Toolkit)** [7]: Una libreria completa per il trattamento del linguaggio naturale. Viene utilizzata per gestire le stopwords, parole comuni che non contribuiscono significativamente all'analisi e che vengono quindi escluse dalla matrice TF-IDF.
- **json** [8]: Una libreria integrata in Python che consente di lavorare con dati in formato JSON. In questo contesto, json viene utilizzata per salvare i dati elaborati in file JSON, rendendoli facilmente accessibili e pronti per ulteriori analisi.
- **pandas** [9]: Una libreria essenziale per la manipolazione e l'analisi dei dati. Viene impiegata per creare e gestire i DataFrame, strutture dati tabellari che semplificano l'analisi delle informazioni estratte e la gestione della matrice TF-IDF.
- **os** [10]: Una libreria integrata in Python che fornisce funzioni per interagire con il sistema operativo. Nel programma viene utilizzata per gestire file e directory, ad esempio per verificare l'esistenza di un file, creare nuove cartelle o gestire percorsi di file.

2.2.1 - Estrazione

La prima operazione effettuata è stata l'estrazione dei post dal file XML. Grazie alla struttura del formato XML, è stato possibile identificare le domande utilizzando l'attributo `post_type`, che quando impostato a 1 indicava una domanda, mentre quando impostato a 2 identificava una risposta. Un passaggio cruciale è stato selezionare la migliore risposta per ciascuna domanda ovvero quella col numero di voti positivi della community. Questo è stato realizzato confrontando i punteggi delle risposte dateci dal tag "score" presente nel file XML, e di conseguenza la risposta con il punteggio più alto è stata considerata la migliore e accoppiata alla rispettiva domanda.

È stato inoltre verificato se le domande avessero una risposta migliore. Nel caso in cui non ci fosse una risposta con voti, ossia se non ci sono risposte al post o se le risposte presenti non hanno ricevuto nemmeno un voto, le domande sono state inserite in una lista dedicata alle domande senza risposta.

2.2.2 - Catalogazione

Per classificare e identificare i post, sono stati adottati due diversi criteri di catalogazione.

Il primo criterio riguarda i post che contengono codice. Per questo, è stata sviluppata una funzione specifica, progettata per restituire esclusivamente i post che includono del codice, sia nella domanda che nella risposta migliore su StackOverflow. La logica alla base di questo criterio è che, se il codice è presente nella domanda, è probabile che si cerchi una risposta contenente del codice di esempio. Allo stesso modo, se il codice è presente nella risposta migliore, si presume che la domanda richieda un esempio pratico sotto forma di codice.

Questa funzione si basa sulla rilevazione dei tag HTML `<code>` e `<pre>`, comunemente utilizzati per formattare e visualizzare il codice all'interno dei post. In questo caso, l'unico parametro preso in considerazione è la presenza di codice sia nelle domande che nelle migliori risposte. Questo significa che i post possono appartenere anche ad altre categorie, come ad esempio domande con un numero di caratteri superiore o inferiore a 700. Tuttavia, l'analisi viene condotta separatamente per ciascuna categoria. Di conseguenza, un post che rientra in più categorie viene analizzato più volte: una volta per la presenza di codice e un'altra per l'equivalenza tra le risposte. Lo scopo di questa catalogazione è analizzare successivamente tramite il programma `aiRequest.py` come ChatGpt generi codice funzionante e sintatticamente corretto in risposta alle domande.

Il secondo criterio di catalogazione riguarda la suddivisione dei post in base alla lunghezza delle domande. In particolare, i post sono stati distinti tra quelli con meno di 700 caratteri e quelli con più di 700 caratteri, questo limite di caratteri è stato deciso in seguito a delle domande al sistema di IA a cui noi facciamo riferimento e che ha espresso essere più propenso a rispondere in modo accurato e preciso a domande con un numero di caratteri non superiore a 700. Questo approccio è stato adottato per valutare se ChatGpt fornisca quindi risposte più accurate e soddisfacenti quando si trova a elaborare domande più brevi, ipotizzando che sia in grado di analizzare e comprendere meglio richieste con un numero inferiore di caratteri.

Infine, è stata creata un'opportunità per l'utente di poter selezionare i post basandosi su specifici termini calcolati tramite TF-IDF [4]. L'algoritmo ha calcolato uno score per ciascun termine presente nelle domande utilizzando la tecnica TF-IDF (Term Frequency - Inverse Document Frequency), che misura l'importanza di un termine in relazione a un documento all'interno di un insieme di documenti. I termini con punteggi TF-IDF elevati, che indicano una maggiore rilevanza rispetto al contesto complessivo, sono stati utilizzati per classificare le domande in base ai termini più rappresentativi. Questo è stato fatto per poter dare un servizio di corrispondenza tra termini e domande per un'analisi concentrata su delle microaree di argomenti.

Il numero limitato di stopwords inizialmente presenti nella libreria `sklearn.feature_extraction` faceva sì che le parole significative potessero essere stopwords. Questo problema è stato risolto ampliando il vocabolario tramite la libreria `nltk`.

Una volta effettuate le catalogazioni sono stati generati tutti i post per ogni tipo di catalogazione nei loro corrispettivi file JSON.

Esempio formattazione output in JSON:

```
{
  "ID": "1059",
  "Question": "I have often heard it said that objects have not delivered in terms of code reuse. Do you agree? If you believe that they haven't, why not? ",
  "Best answer": "No, not necessarily. Objects deliver better semantics, organization of code/functionality and, possibly, ease-of-use. Well designed libraries deliver on the promise of code reuse, not objects per se. "
}
```

2.3 - Elaborazione dei post per catalogazione

Ottenuti tutti i JSON per tutte le rispettive catalogazioni, il prossimo step affrontato è l'elaborazione dei dati.

È stato sviluppato un programma intitolato aiRequest, il cui scopo è raccogliere tutte le diverse tipologie di catalogazione e processarle tramite richieste inviate a ChatGpt tramite API.

In questo programma sono state utilizzate diverse librerie esterne per Python, come:

- **json** [8].
- **os** [10] .

- **time** [12]: Una libreria integrata in Python che fornisce funzioni per la gestione del tempo e dei timer. In questo contesto, time è stata utilizzata per misurare il tempo di esecuzione di determinate operazioni, implementare ritardi temporizzati o calcolare la durata di processi specifici, questo per evitare di superare i limiti di utilizzo di gpt-4o-mini.
- **openai (OpenAI)** [13]: Una libreria fornita da OpenAI che consente di interagire con le API di OpenAI, come GPT. Nel programma, openai è stata utilizzata per inviare richieste ai modelli di OpenAI, come l'ottenimento di risposte generate da GPT, per processare e analizzare il testo.
- **concurrent.futures (ThreadPoolExecutor)** [14]: Una sottolibreria del modulo concurrent.futures di Python che permette di eseguire codice in parallelo utilizzando un pool di thread. In questo contesto, ThreadPoolExecutor è stata utilizzata per gestire l'esecuzione concorrente di più task, migliorando l'efficienza del programma nelle operazioni che possono essere eseguite in parallelo.
- **functools** [15]: Una libreria integrata in Python che fornisce strumenti per la programmazione funzionale. Nel programma, functools è stata utilizzata per decorare funzioni con lru_cache, una funzione che memorizza i risultati di chiamate precedenti per ottimizzare le prestazioni, riducendo i tempi di calcolo per chiamate ripetute.

Il programma ha elaborato i file JSON contenenti tutte le domande e le relative migliori risposte di StackOverflow tramite la funzione principale process_questions. Questa funzione si è occupata di leggere le domande e, in base alla loro classificazione, ha eseguito tre possibili azioni:

- Per le domande che non avevano una migliore risposta, è stata effettuata una richiesta a ChatGpt tramite la funzione ai_answer, una funzione creata per passare a ChatGpt la domanda e restituire una risposta, al fine di fornire una risposta alla domanda che non ne ha avuta.

- Per i post inferiori e superiori a 700 caratteri, è stata effettuata una richiesta a ChatGpt tramite la funzione `ai_answer` per ottenere una risposta generata da ChatGpt. Successivamente, tramite la funzione `compare_answer`, sono state confrontate le due risposte e si è chiesto a ChatGpt se fossero equivalenti.
- Per i post che contenevano codice, è stata sempre effettuata una richiesta a `ai_answer` per ottenere la risposta di ChatGpt ma aggiungendo nella richiesta la stringa: " if there is code in the question, answer with code if needed.". In seguito, è stata richiamata la funzione `code_compiling`, funzione creata per chiedere a ChatGpt di riconoscere il codice presente nelle risposte e verificare se esso compilasse e/o fosse semanticamente corretto rispetto alla domanda posta.

Utilizzo della libreria `time` nella funzione `ai_answer`:

```
start_time = time.time()

global tokens_used, start_time

current_time = time.time()
elapsed_time = current_time - start_time
if elapsed_time < 60:
    if tokens_used + TOKEN_COST_PER_REQUEST >= RATE_LIMIT_TPM -
RATE_LIMIT_BUFFER:
        sleep_time = 60 - elapsed_time
        time.sleep(sleep_time)
        tokens_used = 0
        start_time = time.time()
    else:
        tokens_used = 0
        start_time = current_time
```

Questo è stato fatto per non superare i limiti imposti da openAi sull'utilizzo del modello gpt-4o-mini secondo il sito ufficiale [platforms.openai](https://platform.openai.com/docs/pricing) [16], ovvero quello rispettivo ai token per minuto.

Per le funzioni `code_compiling` e `compare_answer` è stata usata la tecnica dell'autoprompting.

L'autoprompting [5] è una tecnica che sfrutta le capacità dei modelli di GenAI per creare prompt ottimizzati per diverse esigenze. Invece di cercare di formulare manualmente il prompt perfetto, possiamo chiedere di fare una serie di domande mirate, che aiutano a chiarire l'obiettivo e a raffinare il prompt finale.

Questo metodo non solo semplifica il processo di creazione dei prompt, ma può anche migliorare significativamente la qualità delle risposte generate.

Prompt di richiesta per la compilazione/semantica dei codici nella funzione `code_compiling`:

```
def code_compiling(question, ChatGpt_answer, best_answer):
    compile_question = (
        "Say if there is code or not in the question and in the two answers,
        and then say if the code compiles "
        "(i.e., if it's semantically correct) or not for the question and for the
        two answers. "
        "If there is no code, the output should be 'code: No, compile: N/A'. "
        "If there is code, the output should either be 'code: Yes, compile: Yes'
        or 'code: Yes, compile: No'. "
        "Note that 'code: Yes, compile: N/A' is not an acceptable output.\n\n"
        "Output format should be:\n"
        "Question: code: Yes/No, compile: Yes/No/N/A;\n"
        "Answer StackOverflow: code: Yes/No, compile: Yes/No/N/A;\n"
        "Answer ChatGpt: code: Yes/No, compile: Yes/No/N/A.\n\n"
        f"Question: {question}\n"
        f"Answer StackOverflow: {best_answer}\n"
        f"Answer ChatGpt: {ChatGpt_answer}"
    )
    return ai_answer(compile_question, True)
```


Prompt di richiesta per l'equivalenza tra risposte nella funzione `compare_answer`:

```
def compare_answers(ChatGpt_answer, best_answer):  
    comparison_question = f"Are the following two answers  
    equivalent? Say yes or no.\nAnswer 1: {ChatGpt_answer}\nAnswer  
    2: {best_answer}"  
    return ai_answer(comparison_question, False)
```

Dopo aver elaborato tutti i post, sono state inserite le coppie di domande e risposte nel file JSON, complete di verifica di equivalenza, controllo di compilazione e/o correttezza semantica del codice. Questo approccio ha facilitato l'accesso e la produzione dei risultati.

2.4 - Analisi dei risultati

Una volta finita la fase di elaborazione dei dati, è stato creato il programma `resultAnalyzer`. Questo programma è servito per leggere i JSON e ricavare tutte le rispettive risposte sull'equivalenza e sul controllo di compilazione e/o correttezza semantica del codice.

In questo programma sono state utilizzate diverse librerie esterne per Python, come:

- **json** [8].
- **os** [10].
- **matplotlib.pyplot** [18]: Una libreria per la creazione di grafici in Python. Nel programma, `matplotlib` è stata utilizzata per generare visualizzazioni grafiche (bar chart) che mostrano il confronto tra risposte equivalenti e non equivalenti, oltre all'analisi della compilazione del codice.

Come anticipato, nel programma `aiRequest.py`, i dati elaborati sono stati salvati in un file JSON per semplificare l'estrazione e l'analisi dei risultati. Questo approccio facilita l'uso successivo dei dati tramite il programma `resultAnalyzer`.

Per analizzare i risultati, sono state esaminate due principali aree: la compilazione del codice e l'equivalenza delle risposte. In particolare, sono stati calcolati il numero di risposte fornite da ChatGpt che contengono codice e, tra queste, il numero di risposte con codice compilabile, stessa cosa è stata fatta per le risposte fornite da StackOverflow. Inoltre, è stata effettuata un'analisi dell'equivalenza delle risposte, distinguendo tra quelle considerate equivalenti e quelle non equivalenti.

Di seguito degli esempi sui dati analizzati:

Esempio di dato analizzato per le domande contenenti codice:

```
"Answer StackOverflow": {  
  "code": "Yes",  
  "compile": "Yes"  
},  
"Answer ChatGpt": {  
  "code": "No",  
  "compile": "N/A."  
}
```

Da questo esempio, possiamo osservare che per la risposta di StackOverflow è presente del codice, e questo codice è compilabile. Al contrario, la risposta di ChatGpt non include codice, quindi non è stata valutata la compilabilità in questo caso.

Esempio di dato analizzato per le domande con caratteri superiori e inferiori a 700:

"Are the two answers equivalent?": "No."

In questo esempio vediamo come le due risposte analizzate da ChatGpt, secondo esso non sono equivalenti.

Con la raccolta di tutti questi risultati, sono stati creati dei grafici per visualizzare le diverse situazioni e dei file di riepilogo in formato TXT, che descrivono a parole i risultati ottenuti.

Capitolo 3 - Risultati

Una volta analizzati tutti i post e completate le richieste all'IA, sono stati prodotti i risultati per i diversi tipi di catalogazione, principalmente suddivisi in:

- Post con codice.
- Post con un numero di caratteri inferiore e superiore a 700.

Dal dataset iniziale di 1.283 post sono stati ricavati i post contenenti codice pari a 218 post, i post con un numero di caratteri inferiori a 700 pari a 761 post e per quelli superiori a 700 invece 505 post. Mentre 17 post sono i post con le domande senza le migliori risposte.

I risultati sono stati illustrati attraverso grafici ed esempi per facilitarne l'interpretazione.

3.1 - Post con codice

Per i post contenenti codice nelle risposte, è stata valutata l'eseguibilità del codice o, nel caso di frammenti decontestualizzati, la correttezza semantica rispetto alla domanda. L'analisi è stata condotta su una parte del dataset ricavata di 218 domande. Dai risultati è emerso che i post contenenti codice, già selezionati durante la creazione del dataset, sono stati ulteriormente analizzati per la presenza di codice tramite IA e le effettive risposte con codice sono:

- ChatGpt: 94 risposte contenenti codice
- StackOverflow: 79 risposte contenenti codice

Di queste:

- ChatGpt: 92 risposte compilano con successo
- StackOverflow: 56 risposte compilano con successo

Per fornire un esempio testuale dei post analizzati e dei risultati ottenuti, presenteremo degli esempi di post contenenti codice e la relativa analisi.

3.1.1 - Esempi post con codice

3.1.1.1 Esempio di post con codice di StackOverflow semanticamente sbagliato

Domanda:

```
"ID": "937",  
"Question": "I know that they are implemented extremely unsafely in  
C/C++. Can't they be implemented in a safer way? Are the disadvantages  
of macros really bad enough to outweigh the massive power they provide?"
```

Risposta ChatGpt:

```
inline int square(int x) {  
    return x * x;  
}
```

Risposta StackOverflow:

```
#define TWO 1+1  
#define min(x, y) (((x) < (y)) ? (x) : (y))
```

Anche se questo codice è compilabile, può comportarsi in modo inaspettato quando viene utilizzato in espressioni. Pertanto, quando l'IA viene interpellata

sulla sua correttezza semantica, lo considera non semanticamente corretto rispetto a quanto richiesto dalla domanda.

ChatGpt sconsiglia l'uso delle macro in C++ e raccomanda invece di adottare un approccio basato su funzioni inline. Questo codice si compila correttamente ed è in linea con la domanda, poiché propone una soluzione alternativa ma più sicura rispetto a una che potrebbe comportare dei rischi.

Difatti il risultato dell'analisi di compilazione di queste due domande è:

Risultato analisi di compilazione:

```
"Answer StackOverflow": {  
  "code": "Yes",  
  "compile": "No"  
},  
"Answer ChatGpt": {  
  "code": "Yes",  
  "compile": "Yes."  
}
```

3.1.1.2 Esempio di post con codice di ChatGpt semanticamente corretto e compilabile

Domanda:

```
"ID": "566",  
"Question": "goto is almost universally discouraged. Is using this statement  
ever worthwhile? ",
```

Questa domanda pone il problema sull'utilizzo dell'istruzione goto, che è generalmente sconsigliata nella maggior parte dei linguaggi di programmazione e chiede se ci sono casi particolari in cui questa istruzione possa essere vantaggiosa o comunque utile.

Per questo post StackOverflow non risponde tramite del codice.

Risposta di StackOverflow:

"StackOverflow best answer": "This has been discussed several times on StackOverflow, and Chris Gillum summarized the possible uses of goto: Cleanly exiting a function Often in a function, you may allocate resources and need to exit in multiple places. Programmers can simplify their code by putting the resource cleanup code at the end of the function all all \"exit points\" of the function would goto the cleanup label. This way, you don't have to write cleanup code at every \"exit point\" of the function. Exiting nested loops If you're in a nested loop and need to break out of all loops, a goto can make this much cleaner and simpler than break statements and if-checks. Low-level performance improvements This is only valid in perf-critical code, but goto statements execute very quickly and can give you a boost when moving through a function. This is a double-edged sword, however, because a compiler typically cannot optimize code that contains gotos. I'd argue, as many others would argue, that in all of these cases, the usage of goto is used as a means to get out of a corner one coded oneself into, and is generally a symptom of code that could be refactored. ",

Mentre ChatGpt utilizza degli esempi ben formati come:

Risposta ChatGpt:

```
if (allocate_resource() != SUCCESS) { goto cleanup; }
if (another_operation() != SUCCESS) { goto cleanup; }
cleanup:
free_resources();
return error_code;

or

for (int i = 0; i < outer_limit; i++) {
    for (int j = 0; j < inner_limit; j++) {
        if (condition) {
            goto end_loops;
        }
    }
} end_loops:
```

Risultato analisi di compilazione:

```
"Answer StackOverflow": {  
    "code": "No",  
    "compile": "N/A"  
},  
"Answer ChatGpt": {  
    "code": "Yes",  
    "compile": "Yes."  
}
```

3.1.1.3 Esempio di post con errore nella compilazione del codice presente nella domanda a ChatGpt.

Domanda:

```
"ID": "3678",  
"Question": "I work with C# professionally and I write code like this all the  
time.  
private IEnumerable<Something> GetAlotOfSomething() {  
    if (somethingA.IsReady)  
        yield return somethingA;  
    if (somethingB.IsReady)  
        yield return somethingB;  
    if (somethingC.IsReady)  
        yield return somethingC;  
    // ... More complex logic }  
var specialSomethings =    GetAlotOfSomething()  
    .Where(s => s.IsSpecial);  
Then one day I have to write a bit of VB6 or JScript and I end up writing so  
much boilerplate just to get things done. Anyone thoughts? ",
```

La domanda chiede un'opinione o suggerimenti su come affrontare il problema di dover scrivere codice "boilerplate" quando si passa da un linguaggio moderno

come C#, che offre funzionalità avanzate come yield return e LINQ, a linguaggi più vecchi come VB6 o Jscript.

Risposta ChatGpt:

```
Function IsReady(something As SomethingType) As Boolean
    Return something.IsReady
End Function

If somethingA.IsReady Or somethingB.IsReady Or somethingC.IsReady
Then ' Your processing logic here
End If

var specialSomethings = allSomethings.filter(function(s) {
    return s.IsSpecial;
});
```

La risposta di ChatGpt è pertinente e semanticamente corretta rispetto alla domanda e compilabile, presenta degli esempi concreti di come gestire il codice in linguaggi come VB6 o Jscript.

Risposta StackOverflow:

"StackOverflow best answer": "Iterators (generators etc) are certainly great features that I use a lot. I don't qualify them as necessary, but I will certainly choose languages that have them when I get a choice. ",

Mentre StackOverflow non fa uso di codice, ma descrive l'utilità degli iteratori ma non affronta il problema centrale della domanda.

Risultato analisi di compilazione:

```
"Answer StackOverflow": {  
  "code": "No",  
  "compile": "N/A"  
},  
"Answer ChatGpt": {  
  "code": "Yes",  
  "compile": "No."  
}
```

Possiamo vedere come per StackOverflow non è presente codice, cosa giusta, mentre per ChatGpt il codice è presente ma non compila o/e semanticamente non corretto rispetto alla domanda. Questo risultato non è corretto e quindi abbiamo la presenza di un errore nell'analisi delle domande dei post con codice.

3.1.1.4 Esempio di post con errore di codice non compilabile nella risposta di StackOverflow

Domanda:

```
"ID": "7536",  
"Question": "Possible Duplicate: int* i; or int *i; or int * i; Thinking about  
where we place our asterisks; how do those that prefer to keep the  
\"pointerness\" away from the type and with the identifier (int *i) write  
code when the identifier is missing? void f(int*); // 1 void f(int *); // 2 The  
former seems much more common, no matter what your preference when  
with the identifier. Is this a special case? What makes it an exception?  
However, the first still isn't universal, because I have seen the latter  
style. Besides consistency along the lines of \"there's always a space with  
the identifier, so we have one without\", are there any other reasons to  
prefer it? What about casts or array and function types? How would you  
re-write these: (void*)var /*or*/ (void *)var int[3] /*or*/ int [3] // more  
relevant in C++ than C: Example<int[3]> void(int) /*or*/ void (int) // more  
relevant in C++ than C: std::function<void(int)> The latter two would  
rarely, if ever, be used in C, but are seen with C++ templates. ",
```

La domanda sta chiedendo un'opinione sullo stile di posizionamento dell'asterisco (*) nelle dichiarazioni di puntatori in C/C++.

Risposta ChatGpt:

ChatGpt non usa del vero e proprio codice ma più una spiegazione sulla sintassi corretta seguita da dei mini esempi non considerabili come pezzi di codice.

Risposta di StackOverflow:

```
void f(int *);  
void f(int *x) {  
    char *y = (char *) x;  
    char x[4]; }
```

La definizione della funzione è corretta, il problema sta nell'uso della variabile x due volte. Una volta come parametro e una volta come variabile locale. Questo viene interpretato come errore di semantica dove la soluzione sarebbe quella di rinominare almeno una delle due variabili. Questo è quello che ChatGpt riesce a capire da questo codice senza tener conto quindi che il risultato di x lo salva in y e quindi x risulta sovrascrivibile senza nessun problema

Risultato analisi di compilazione:

```
"Answer StackOverflow": {  
    "code": "Yes",  
    "compile": "No"  
},  
"Answer ChatGpt": {  
    "code": "No",  
    "compile": "N/A."  
}
```

Difatti nel risultato dell'analisi di compilazione troviamo che il codice per la risposta di StackOverflow è presente ma non risulta compilabile a causa dell'errore di semantica visto prima e che in realtà non sussiste.

3.2 - Post con caratteri superiori e inferiori a 700

Per i post che non contengono codice, sono state raggruppate diverse categorie, tra cui:

- Post con un numero di caratteri superiore a 700.
- Post con un numero di caratteri inferiore a 700.

Sono state analizzate le equivalenze tra le migliori risposte di StackOverflow e le risposte di ChatGpt per tutti questi tipi di post. L'analisi è stata condotta sull'intero dataset di 1.283 domande e dai risultati è emerso che:

- 1261 domande non avevano risposte equivalenti.
- 5 domande avevano risposte equivalenti.

Per i post con meno di 700 caratteri, solo 3 risposte su 761 risultano equivalenti, mentre per i post con più di 700 caratteri, solo 2 risposte su 505 sono equivalenti. Questo dato non supporta l'ipotesi iniziale secondo cui, con meno caratteri, ChatGpt avrebbe offerto risposte più precise e meno generiche e quindi un numero alto di risposte equivalenti nei post con caratteri inferiori a 700 rispetto a quelli superiori a 700.

3.2.1 - Esempi post con caratteri inferiori e superiori a 700.

Per illustrare il tipo di analisi effettuata e i risultati ottenuti, presenteremo degli esempi con dei post inferiori a 700 caratteri per motivi di brevità e semplicità.

3.2.1.1 *Esempio di post con risposte non equivalenti*

Domanda:

"ID": "16445",
"Question": "Where I work, employees use a third-party desktop program for their clients. This program saves data to a flat file. My colleague wants to write a Java program that uploads that flat file to a remote server, opens the desktop program when the flat file is downloaded from a Web site, and checks if the desktop program is running or not by looking at the Windows processes. He keeps calling this helper/utility program a \"wrapper.\" But it doesn't wrap anything! I tried to clear it up with him, but he said, \"Well, I call it a wrapper.\" He now has everyone in the company calling it a \"wrapper.\" What would you call it? I say that it's a helper program or utility program. "

Risposta di ChatGpt:

"ChatGpt answer": "You're correct in suggesting that the term \"wrapper\" may not be the most accurate description for the program your colleague wants to create. Typically, a \"wrapper\" refers to software that encapsulates or interfaces with another underlying program or library, often providing a simpler or enhanced interface. In the case you've described, the Java program doesn't seem to be wrapping another piece of software but rather performing a variety of tasks uploading files, launching an application, and checking processes. Given its functionality, you could accurately describe it as a \"utility program\" or a \"helper program.\" These terms more clearly convey the program's role as a tool that assists users in performing specific tasks instead of encapsulating or modifying another program. You might also consider terms like \"automation script\" or \"file management tool,\" depending on how your colleague envisions the program functioning and its primary purpose. Whatever you decide to call it, highlighting its utility may help clarify its purpose within the company."

Risposta di StackOverflow:

"StackOverflow best answer": "I don't think there's an official definition! In my mind, a wrapper program is a program that, in general, starts before the wrapped program starts, and exits no earlier than after the wrapped program exits. A wrapper program would probably also manage the lifetime of the wrapped program (start it for you, stop it for you.) I recommend against arguing too much over the names of things, though. The important thing is to design software that works well. Sometimes belaboring terminology can aid in this goal; sometimes it is a hobgoblin. "

La risposta di ChatGpt è più dettagliata e orientata a trovare una terminologia più precisa per descrivere il programma specifico, mentre la risposta di StackOverflow è più generale e focalizzata sull'aspetto funzionale del software. Le due risposte non sono equivalenti poiché adottano approcci e prospettive differenti.

Difatti questo è il risultato dato:

Risultato analisi di equivalenza:

"Are the two answers equivalent?": "No."

3.2.1.2 Esempio di post con risposte equivalenti

Risposta di StackOverflow:

```
"ID": "15360",  
"Question": "I read this answer and found a comment insisting not to send  
password by email: passwords should not be able to be retrieved by email,  
I hate that. It means my password is stored in plain text somewhere. it  
should be reset only. This raises me the question of handling Forgot  
Password option? At any cost the raw password must be displayed in any  
UI so that user will be able to read it. So what would be the way to handle  
\"Forgot Password\" ",
```

La domanda chiede quale sia il modo corretto di gestire la funzionalità "Forgot Password" in un'applicazione web. In particolare, l'autore esprime preoccupazioni riguardo alla sicurezza dei sistemi che permettono di recuperare le password via email, poiché ciò implica che la password sia conservata in chiaro da qualche parte, il che è considerato un rischio di sicurezza.

Risposta di StackOverflow:

```
"StackOverflow best answer": "A good application design will not be able  
to explicitly recover a users password. This is because it is usually stored  
after it is run through some sort of hash which is a one way operation. The  
best way to handle lost password is to perform a reset, email to the users  
account a link with a generated parameter tacked on that identifies this as  
a valid password reset for the account in question. At this point they can  
set a new password. This does assume you have a users email address on  
file. ",
```

La risposta di StackOverflow spiega che una buona applicazione non dovrebbe mai recuperare esplicitamente la password dell'utente, poiché le password dovrebbero essere memorizzate in un formato non reversibile.

Risposta ChatGpt:

"ChatGpt answer": "You're correct in pointing out that passwords should never be stored in plain text or sent in an email. Instead, the best practice for handling the \"Forgot Password\" functionality is to use a secure password reset process. Here how it typically works: 1. **Password Reset Request**: When a user indicates that they've forgotten their password (e.g., by clicking a \"Forgot Password?\" link), you should prompt them to enter their registered email address. 2. **Generate a Reset Token**: After the user submits their email address, generate a unique, time-limited token (e.g., a cryptographically secure random string) and store it in your database along with the user's profile. This token is distinct from the user's password. 3. **Send Reset Email**: Send an email to the user with a link that includes the token as a URL parameter. The email should inform the user that they have requested a password reset and provide instructions to click on the link to reset their password. 4. **Create a Password Reset Form**: The link in the email should direct the user to a secure webpage (the password reset form) where they can input a new password. When they access this page, you should validate the token to ensure it is still valid (not expired and corresponds to that user). 5. **Update Password**: Once the user submits the new password via the form, hash the new password (using a strong hashing algorithm, e.g., bcrypt) and store it in the database. In addition, invalidate the reset token to prevent it from being reused. 6. **Confirmation**: After successfully updating the password, you can redirect the user to a login page or display a confirmation message indicating that their password has been successfully reset. ### Key Points to Remember: - **Do Not Send Passwords**: Never send the actual password via email. Instead, use the reset token mechanism. - **Securely Hash Passwords**: Always hash passwords before storing them. - **Token Expiry**: Ensure that the reset token has an expiration time (e.g., 1 hour) to enhance security. - **Logging and Monitoring**: Log password reset requests to monitor for potentially malicious activity. - **Rate Limiting**: Implement rate limiting on the password reset requests to prevent abuse (e.g., brute-force attacks). By following this process, you can provide a secure and user-friendly way for users to recover their accounts without compromising their password security."

La risposta di ChatGpt fornisce una guida dettagliata su come gestire in modo sicuro la funzionalità "Forgot Password". Raccomanda di non inviare mai password in chiaro via email. Invece, suggerisce di utilizzare un processo di reset della password basato su un token univoco e temporaneo.

Di conseguenza possiamo concludere che anche se le due risposte hanno magari delle spiegazioni diverse e metodi di scrittura differenti, le due risposte si basano sullo stesso principio ovvero quello di non recuperare la password in chiaro.

Difatti il risultato tra queste due risposte è:

Risultato analisi di equivalenza:

"Are the two answers equivalent?": "Yes."

3.2.1.3 Esempio di post con errore nell'equivalenza delle risposte

Durante l'analisi dei risultati per i post con le risposte equivalenti non si sono riscontrati delle false equivalenze tra risposte. Infatti tutte le risposte segnate come equivalenti lo sono, almeno nel concetto di base della risposta anche se possono differire sul livello di dettaglio.

3.2.1.4 Esempio post con risposte non equivalenti

Domanda:

"ID": "5560",
"Question": "It's all the rage nowadays. \"Everyone\" recommends it. That in and of itself makes me suspicious. What are some disadvantages you have found when doing test-first (test-driven) development? I'm looking for personal experiences from knowledgeable practitioners--I can read the hypothetical musings of a hundred wannabes elsewhere on the internet. I ask not because I am looking to hate TDD, but because it is my job to improve software development process, and the more we can learn about the problems people encounter, the better chance we have of improving the process. ",

Risposta StackOverflow:

"Stack Overflow best answer": "There are quite a few, but the advantages far outweigh the disadvantages. There's a steep learning curve. Many developers seem to expect that they can be efficient with test-first programming right from day one. Unfortunately it takes a lot of time to gain experience and program at the same speed as before. You can't get around it. To be more specific, it's very easy to get wrong. You can very easily (with very good intentions) end up writing a whole bunch of tests which are either difficult to maintain or testing the wrong stuff. It's difficult to give examples here - these kind of issues simply take experience to solve. You need to have a good feel of separating concerns and designing for testability. My best advice here would be to do pair-programming with someone who knows TDD really well. You do more coding up front. Test-first means you can't skip tests (which is good) and means you'll end up writing more code up front. This means more time. Again, you can't get around it. You get rewarded with code that's easier to maintain, extend and generally less bugs, but it takes time. Can be a tough sell to managers. Software managers are generally only concerned with timelines. If you switch to test-first programming and you're suddenly taking 2 weeks to complete a feature instead of one, they're not gonna like it. This is definitely a battle worth fighting and many managers are enlightened enough to get it, but it can be a tough sell. Can be a tough sell to fellow developers. Since there's a steep learning curve not all developers like test-first programming. In fact, I would guess that most developers don't like it at first. You can do things like pair-programming to help them get up to speed, but it can be a tough sell. In the end, the advantages outweigh the disadvantages, but it doesn't help if you just ignore the disadvantages. Knowing what you're dealing with right from the start helps you to negotiate some, if not all, of the disadvantages. ",

La risposta di Stack Overflow è più concisa e si concentra su problemi pratici e difficoltà immediate, come la curva di apprendimento e la difficoltà di vendere TDD a manager e colleghi.

Mentre la risposta di ChatGpt risulta più estesa ma anche quindi più dettagliata:

Risposta di ChatGpt:

"ChatGpt answer": "Test-Driven Development (TDD) has several challenges and disadvantages despite its advocates. Teams new to TDD may face a significant learning curve, with initial difficulties slowing down development. In the beginning, it may seem that more time is spent writing tests than developing features, which can lead to frustration and skepticism. TDD can also lead to an overemphasis on unit tests at the expense of integration or end-to-end tests, creating a false sense of security about software quality. Maintaining tests can become burdensome, with the risk that they become outdated and require significant time to update, which may discourage writing new tests. The challenges increase when requirements are unclear or subject to frequent changes, resulting in wasted effort and test rewrites. Applying TDD to legacy code is difficult and can lead to frustration as it wasn't originally designed with testing in mind."

La risposta di ChatGpt è stata riassunta per renderla visualizzabile all'interno del documento ma analizzandola essa fornisce un'analisi più dettagliata e amplia il campo degli svantaggi di TDD, coprendo vari aspetti pratici e psicologici che possono impattare il processo di sviluppo. Fornisce anche suggerimenti specifici su come mitigare questi problemi.

Di conseguenza il risultato dell'analisi è:

Risultato analisi di equivalenza:

"Are the two answers equivalent?": "No."

La risposta di ChatGPT sembra seguire meglio lo scopo ultimo della domanda, offrendo una panoramica più ampia e dettagliata delle sfide pratiche e psicologiche legate a TDD. Fornisce anche spunti per affrontare e mitigare questi svantaggi, il che è utile per migliorare il processo di sviluppo software. La risposta

di Stack Overflow è utile e pratica ma meno dettagliata, concentrandosi maggiormente sugli aspetti pratici e le difficoltà immediate nell'adozione di TDD.

3.2.1.5 Esempio post con risposte non equivalenti

Domanda:

"Question": "Need a Free, Fast(development and runtime) and Reliable(transactions and locking) tool set for creating an enterprise application for SMBs. I'm thinking of an application framework + UI framework + DB which will help me in developing the software faster. As it is known, business softwares need lot of similar UIs to be created. My idea is ...to create a new form with N number of fields and connecting it to the database for basic CRUD operations within 30min to 1hr. I got the taste of Intersystems Cache technology stack with an app framework on top of it. To be frank...it is amazing... I'm looking out for something similar to this in opensource. Any suggestions? ",

La domanda chiede suggerimenti per un set di strumenti gratuito, veloce (sia nello sviluppo che nell'esecuzione) e affidabile (in termini di transazioni e locking) per creare un'applicazione enterprise destinata a piccole e medie imprese (PMI).

Risposta StackOverflow:

"Stack Overflow best answer": " What is the best toolset for living a fullfilling life for suburban professionals? Need a convenient, functional toolset for living my life. I'm thinking of a moral code + religion + culture which will help me life happier and better. Ok I couldn't help myself. You're asking for religion and any answer you get will be based more on personal preferences than one being actually \"better\" than another. That being said, go with Ruby. It's free, fully functional and comes with a lot of stuff, like Rails and an active community. And, you get to look down on everyone else as a bonus. My background is in Java and .NET, but if I had to start from scratch at this moment, I'd probably learn Ruby. ",

La risposta di StackOverflow discute il miglior set di strumenti da una prospettiva più personale, paragonandolo a scelte di vita e codici morali e suggerisce Ruby.

Risposta di ChatGpt:

"ChatGpt answer": "For developing enterprise applications for small to medium-sized businesses quickly and reliably, you can consider several open-source technology stacks. Options include using Spring Framework with Thymeleaf and PostgreSQL, Django with Django REST Framework and SQLite/PostgreSQL, Ruby on Rails with Hotwire and PostgreSQL/MySQL, Flask with Flask-WTF and SQLite/PostgreSQL, or .NET Core with Blazor and SQL Server or SQLite. Additionally, for low-code solutions, Budibase and Appsmith provide platforms for rapid app development with minimal coding. Each of these options supports efficient development while ensuring robust backend capabilities for transactions and data integrity."

La risposta di ChatGpt è stata riassunta per renderla visualizzabile nel documento, fornisce suggerimenti dettagliati su stack tecnologici gratuiti, veloci e affidabili per sviluppare applicazioni aziendali per le PMI, elencando specifici stack tecnologici con spiegazioni, tra cui il Framework Spring con Thymeleaf e PostgreSQL, Django con Django REST Framework e SQLite/PostgreSQL, Ruby on Rails con Hotwire e PostgreSQL/MySQL, Flask con Flask-WTF e SQLite/PostgreSQL, e .NET Core con Blazor e SQL Server/SQLite. Include anche suggerimenti bonus per piattaforme low-code come Budibase e Appsmith.

Risultato analisi di equivalenza:

"Are the two answers equivalent?": "No."

Questo perché la risposta di ChatGPT fornisce una guida dettagliata e pratica, con molti suggerimenti e esempi di piattaforme mentre la risposta di Stack Overflow è più personale e informale. Inoltre anche le piattaforme suggerite sono diverse.

3.3 - Catalogazioni per TF-IDF

Come anticipato nel capitolo metodologia, è stata sviluppata una funzionalità per la raccolta di post per un termine specifico.

3.3.1 - Esempi di post classificati secondo il termine “Programming”

Per portare un esempio della riuscita metodologia di catalogazione secondo TF-IDF, di seguito troviamo degli esempi riportati nella catalogazione per il termine “Programming”.

Domanda:

```
"ID": "6974",  
"Question": "What's the worst programming-related or technical book  
you've ever read? This can be any book which one way or another can be  
connected to programming, as long as it's not fiction. ",
```

Questa domanda, poiché chiede di un libro legato alla programmazione o a un argomento tecnico, anche se si tratta di un'esperienza negativa rientra nella possibile catalogazione “Programming”, in quanto il focus è comunque su contenuti tecnici o di programmazione.

Domanda:

```
"ID": "9006",  
"Question": "Functional programming is one of the oldest programming  
paradigms. However it isn't used much in the industry compared to more  
popular paradigms. But it have largely been emphasized in academia.  
What's your strongest opinion against functional programming? ",
```

Pure questa domanda rientrerebbe nella sezione programming in quanto riguarda un argomento direttamente correlato ai paradigmi di programmazione, in particolare la programmazione funzionale. La domanda chiede opinioni negative o critiche su questo paradigma, il che è un tema rilevante per programmatori e sviluppatori che lavorano o studiano diverse tecniche di programmazione.

Questo tipo di classificazione secondo un termine specifico ha dato dei buoni risultati, anche se per alcune domande non ha ben funzionato.

3.4 - Considerazioni sui risultati

L'analisi dei dati raccolti ha fornito un quadro chiaro delle differenze tra le risposte fornite da StackOverflow e quelle generate da ChatGpt, evidenziando per ogni fonte punti di debolezza e forza.

3.4.1 - Considerazioni risultati dei post con codice

L'analisi dei post contenenti codice ha mostrato che ChatGpt è in grado di generare codice compilabile con un'alta percentuale di successo, raggiungendo un tasso del 97,87% (92 su 94 istanze). Questo risultato è notevole, considerando che il codice è generato in modo automatico e senza un contesto approfondito rispetto alla

domanda. Tuttavia, una critica è necessaria sulla correttezza semantica del codice prodotto. Sebbene il codice di ChatGpt tenda a essere sintatticamente corretto, la sua validità semantica può variare. In alcune istanze, il codice potrebbe compiere operazioni tecnicamente corrette ma non rispondere in modo preciso alla richiesta dell'utente, potenzialmente introducendo bug o comportamenti inattesi. Al contrario, le risposte di StackOverflow, pur avendo un tasso di compilazione inferiore pari al 70,88% (56 istanze su 79), beneficiano del contributo diretto di programmatori esperti che spesso tengono conto del contesto specifico, dei requisiti funzionali e delle best practice di programmazione, producendo risposte più aderenti alle esigenze degli utenti.

Inoltre, i post di StackOverflow tendono a includere spiegazioni dettagliate e discussioni sui pro e contro delle diverse soluzioni, offrendo agli utenti non solo una risposta immediata, ma anche una comprensione più profonda del problema e delle possibili soluzioni. Questo tipo di interazione è cruciale per l'apprendimento e per l'adozione di pratiche di programmazione migliori.

3.4.2 - Considerazioni risultati dei post con caratteri inferiori e superiori a 700

L'analisi dei post con caratteri superiori e inferiori a 700, ha rivelato una differenza significativa tra le risposte generate da ChatGpt e quelle presenti su StackOverflow. Su 1402 domande analizzate, solo 5 hanno mostrato risposte equivalenti, evidenziando una scarsa sovrapposizione tra le due fonti. Si è osservato che le risposte di ChatGPT tendono ad essere più precise e dettagliate rispetto a quelle di StackOverflow, che risultano più generiche e con spesso delle considerazioni del tutto personali, cosa che rende StackOverflow molto forte per la risoluzione di problemi. Inoltre, si è rilevato che le due catalogazioni effettuate non hanno prodotto i risultati sperati, poiché sia per le domande con meno di 700 caratteri

sia per quelle con più di 700 caratteri, le equivalenze registrate sono state poche e senza uno scarto importante.

Un aspetto cruciale è lo stile delle risposte di ChatGpt, che tende a essere uniforme, con una struttura discorsiva che abbraccia vari aspetti della domanda. Questo approccio è utile per fornire una panoramica generale, ma può risultare meno efficace quando è necessaria una risposta più precisa e focalizzata. Al contrario, le risposte su StackOverflow sono generalmente più concise e dirette, concentrate sull'aspetto pratico della soluzione ma su considerazioni e deduzioni molto personali al contrario di ChatGpt.

3.4.3 - Considerazioni per la catalogazione secondo TF-IDF

I risultati ottenuti per questo tipo di catalogazione hanno fornito un quadro chiaro: il sistema di catalogazione funziona molto bene. Sebbene non possiamo garantire un successo del 100% utilizzando un algoritmo TF-IDF, i risultati indicano una buona precisione nella categorizzazione.

3.4.4 - Considerazioni generali

I risultati quindi suggeriscono che, mentre ChatGpt è uno strumento potente per generare rapidamente risposte sintatticamente corrette e fornire spiegazioni articolate, l'affidabilità e l'utilità delle sue risposte possono essere limitate dalla mancanza di contesto e di esperienza pratica che caratterizza invece le risposte degli utenti di StackOverflow. La collaborazione tra l'IA e gli esperti umani potrebbe rappresentare un'area di grande potenziale, combinando la velocità e l'accessibilità di ChatGpt con l'affidabilità delle risposte fornite da una comunità di esperti.

Per i post con codice, ChatGpt può servire come punto di partenza rapido per generare soluzioni o per ottenere spiegazioni tecniche di base, mentre StackOverflow può essere utilizzato per approfondire la discussione, valutare diverse soluzioni e scegliere l'approccio più adatto al problema specifico. Per i post senza codice, l'utilizzo di ChatGpt può essere utile per ottenere una risposta molto ben articolata e spesso anche corretta, mentre StackOverflow rimane una risorsa indispensabile per ricevere consigli pratici e basati sull'esperienza.

In conclusione, mentre l'intelligenza artificiale ha dimostrato potenzialità promettenti, la sua integrazione con il contributo umano rimane essenziale per garantire soluzioni tecniche di alta qualità e pertinenti alle esigenze degli utenti. L'interazione sinergica tra questi strumenti può migliorare notevolmente l'esperienza degli utenti nella risoluzione di problemi complessi nel campo dell'informatica.

Capitolo 4 - Conclusioni

L'obiettivo principale di questo lavoro era confrontare le risposte fornite dagli utenti di StackOverflow con quelle generate da un modello avanzato di intelligenza artificiale, come ChatGpt, per valutare l'efficacia di queste tecnologie nel fornire soluzioni tecniche di alta qualità per i programmatori.

Abbiamo analizzato come StackOverflow, basato sul contributo collaborativo della comunità, offra risposte elaborate e validate da esperti, mentre ChatGpt fornisce risposte immediate, flessibili e adattabili a una vasta gamma di richieste.

I risultati ottenuti hanno evidenziato l'utilità di queste due tecnologie nel miglioramento del settore della programmazione. In particolare, abbiamo osservato che ChatGpt si dimostra efficace nel fornire risposte generali e esaustive a domande testuali riuscendo a essere molto completo e generando del testo ben formato. Grazie alla sua capacità di offrire una panoramica ampia, l'utilizzo dell'intelligenza artificiale, sotto la supervisione di un esperto, appare una soluzione ottimale per questo tipo di richieste. È emerso che non vi è alcuna differenza significativa tra le domande con un numero di caratteri inferiore a 700 rispetto a quelle più lunghe, un risultato inaspettato. È possibile, quindi, che questo apparente limite delle IA sia destinato a evolversi nel tempo e che stiamo assistendo all'inizio della sua risoluzione.

Per quanto riguarda le domande contenenti codice, ChatGpt è in grado di generare soluzioni semplici e funzionanti. Anche se il codice prodotto da ChatGpt è sempre compilabile e quindi sintatticamente corretto, non garantisce sempre la risoluzione efficace del problema. Spesso, il codice generato potrebbe non rispondere completamente all'obiettivo della domanda o potrebbe richiedere modifiche significative per funzionare come previsto.

In sintesi, possiamo affermare che ChatGpt tende a essere molto discorsivo per tutti i tipi di domande. L'utilizzo di esso senza la supervisione di un programmatore esperto potrebbe portare a risposte formalmente corrette, ma che spesso non riescono a risolvere il problema nel suo nucleo.

Quindi la metodologia migliore sarebbe quella di affiancare queste due risorse, consentendo così di ottimizzare il processo di apprendimento e problem-solving, combinando la ricchezza delle esperienze umane con la rapidità e l'accuratezza dell'intelligenza artificiale. Questa sinergia offre un notevole vantaggio nel mondo della programmazione, permettendo di affrontare problemi complessi in modo più efficiente e di accelerare lo sviluppo di soluzioni innovative.

In un futuro neanche troppo remoto si potrebbe pensare di ripetere questo studio. Con l'evoluzione tecnologica e le continue migliorie che l'IA acquisirà nel corso degli anni, è plausibile che i risultati cambino, rivelando che il coinvolgimento di un esperto potrebbe diventare superfluo, rendendo l'IA sempre più autonoma ed efficace nel risolvere problemi complessi senza una supervisione umana.

Bibliografia

[1] [Online].

Available: <https://stackoverflow.com/>.

[2] [Online].

Available: <https://ChatGpt.com/>. [Consultato il giorno 7 September 2024].

[3] S. Exchange, 21 January 2014. [Online].

Available: <https://archive.org/details/stackexchange>.

[4] [Online].

Available: <https://pypi.org/project/lxml/>.

[5] [Online].

Available: <https://pypi.org/project/beautifulsoup4/>.

[6] [Online].

Available: <https://pypi.org/project/scikit-learn/>.

[7] [Online].

Available: <https://pypi.org/project/nltk/>.

[8] [Online].

Available: <https://pypi.org/project/jsonlib/>.

[9] [Online].

Available: <https://pypi.org/project/pandas/>.

[10] [Online].

Available: <https://docs.python.org/3/library/os.html#module-os>.

[11] A. Aizawa, «45-65,» 2003. [Online].

Available: <https://www.sciencedirect.com/science/article/pii/S0306457302000213>.

[12] [Online].

Available: <https://docs.python.org/3/library/time.html>.

[13] [Online].

Available: <https://pypi.org/project/openai/>.

[14] [Online].

Available: <https://docs.python.org/3/library/concurrent.futures.html#concurrent.futures.ThreadPoolExecutor>.

[15] [Online].

Available: <https://docs.python.org/3/library/functools.html#module-functools>.

[16] [Online].

Available: <https://platform.openai.com/settings/organization/limits>.

[17] 9 June 2024. [Online].

Available: <https://www.thesundayprompt.com/>

[autoprompting-facciamoci-aiutare-direttamente-da-ChatGpt/](#).

[18] [Online].

Available: <https://pypi.org/project/matplotlib/>.