



Università
di Catania

DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA TRIENNALE IN INFORMATICA

Lorenzo Maria Alberto Paoria

Un analisi tra Stack Overflow e ChatGpt nella risoluzione di problemi di Programmazione

RELAZIONE PROGETTO FINALE

Relatore: Prof. Emiliano Tramontana

Correlatori:
Prof. Gabriella Verga
Dott. Alessandro Midolo

Anno Accademico 2023 – 2024

Indice

[illegible]

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'obiettivo principale è analizzare la pertinenza, l'accuratezza e l'affidabilità delle risposte prodotte dall'intelligenza artificiale rispetto a quelle offerte da esperti umani, 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:

- **Stack Overflow** [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:** Verrà illustrato l'approccio adottato per ottenere i risultati, descrivendo in dettaglio le tecniche e gli strumenti utilizzati nel processo di analisi.
- **Risultati:** Saranno presentati i risultati ottenuti, arricchiti da esempi pratici che ne facilitano la comprensione e dimostrano l'applicazione concreta.
- **Conclusione:** I risultati saranno interpretati ed analizzati per estrapolarne il significato, fornendo una sintesi delle implicazioni emerse e delle possibili prospettive future.

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.

2.1 - Dataset StackOverflow

Per prima cosa, e sicuramente la più importante, ho acquisito un dataset contenente i post del sito Stack Overflow[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 Stack Overflow.

Il dataset si presenta sotto forma di un file XML, contenente sia tutte le domande che tutte le risposte correlate. È importante sottolineare che il file include non solo le risposte migliori, ma tutte le risposte fornite dagli utenti. Questo aspetto è cruciale per la nostra analisi, poiché il nostro obiettivo è concentrarci esclusivamente sulle risposte migliori, quelle con il punteggio più alto, per garantire un'analisi più precisa e mirata.

2.2 - dataCatalog.py

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. 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**: 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 Stack Overflow e estrarre le informazioni rilevanti dai post, come il testo delle domande e delle risposte.
- **BeautifulSoup (bs4)**: 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**: 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)**: 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**: 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**: 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**: 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. Questo è stato realizzato confrontando i punteggi delle risposte: la risposta con il punteggio più alto è stata considerata la migliore e accoppiata alla rispettiva domanda.

Inoltre, è stato verificato se le domande avessero delle risposte migliori. Qualora non le avessero, 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 contenenti codice. A tal fine, è stata sviluppata una funzione specifica, progettata per restituire esclusivamente i post che includono del 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: ciò significa che i post potrebbero appartenere anche ad altre categorie, come domande con un numero di caratteri superiore o inferiore a 700. Lo scopo di questa catalogazione è analizzare 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 approccio è stato adottato per valutare se ChatGPT fornisca 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.

Un problema riscontrato in questo algoritmo, dovuto all'utilizzo della libreria `sklearn.feature_extraction`, è stato il vocabolario limitato delle stopwords. Questo problema è stato risolto ampliando il vocabolario tramite la libreria `nltk`.

Una volta effettuate tutte queste 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 - aiRequest.py

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

È stato sviluppato un programma chiamato `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**: Una libreria integrata in Python che consente di lavorare con dati in formato JSON. Nel programma, json è stata utilizzata per leggere e scrivere dati in file JSON, facilitando la serializzazione e deserializzazione dei dati strutturati.
- **os**: Una libreria integrata in Python che fornisce funzioni per interagire con il sistema operativo. Nel programma, os è stata utilizzata per gestire file e directory, come verificare l'esistenza di file, creare nuove cartelle e gestire percorsi di file in modo portabile tra diversi sistemi operativi.
- **time**: 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.
- **openai (OpenAI)**: 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)**: 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**: 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, al fine di fornire una risposta alla domanda.
- 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, chiedendo a ChatGPT di riconoscere il codice presente nelle risposte e verificare se esso compilasse e/o fosse semanticamente corretto rispetto alla domanda posta.

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:

```
"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}"
```

Prompt di richiesta per l'equivalenza tra risposte:

```
f"Are the following two answers equivalent?"
"Say yes or no."
"\nAnswer 1: {chatgpt_answer}"
"\nAnswer 2: {best_answer}"
```

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 - resultAnalyzer.py

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**: Una libreria integrata in Python che consente di lavorare con dati in formato JSON. Nel programma, json è stata utilizzata per leggere e scrivere dati in file JSON, facilitando la serializzazione e deserializzazione dei dati strutturati.
- **os**: Una libreria integrata in Python che fornisce funzioni per interagire con il sistema operativo. Nel programma, os è stata utilizzata per gestire file e directory, come verificare l'esistenza di file, creare nuove cartelle e gestire percorsi di file in modo portabile tra diversi sistemi operativi.
- **matplotlib.pyplot**: 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 Stack Overflow 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.

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 un dataset di 218 domande. Dai risultati è emerso che i post contenenti codice, già selezionati durante la creazione del dataset, erano:

- ChatGpt: 94 istanze
- StackOverflow: 79 istanze

Di queste:

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

"Stack Overflow best answer": "This has been discussed several times on Stack Overflow, 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 di 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:

"Stack Overflow 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 Stack overflow non fa uso di codice, ma descrive l'utilità degli iteratori ma non affronta il problema centrale della domanda.

Dall'analisi di compilazione di queste due domande il risultato è parzialmente giusto.

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 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.

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 presenta ma non risulta compilabile a causa dell'errore di semantica visto prima.

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 limitChar.
- Post con un numero di caratteri inferiore a limitChar.

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 su un dataset di 1402 domande e dai risultati è emerso che:

- 1397 domande non avevano risposte equivalenti.
- 5 domande avevano risposte equivalenti.

In particolare per i post con caratteri inferiori a 700 solo 3 risposte su 761 sono equivalenti, mentre per i post con caratteri superiori a 700 solo 2 risposte su 505 sono equivalenti.

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

Per illustrare il tipo di analisi effettuata e i risultati ottenuti, presenteremo un esempio che include una domanda con due risposte equivalenti.

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\u2014uploading 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\u2019s 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 Stack Overflow:

"Stack Overflow 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 Stack Overflow 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."

3.3 - 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.

3.3.1 - 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.

3.3.2 - 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. Questo indica che le risposte di ChatGPT tendono a essere più generiche e meno mirate ai dettagli specifici della domanda, mentre quelle di StackOverflow sono spesso più focalizzate allo scopo ultimo della domanda. 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. Grazie alla natura collaborativa della piattaforma, le risposte vengono affinate attraverso un processo iterativo di feedback da parte della comunità, portando alla costruzione di soluzioni concrete e ben definite. Questo stile si dimostra particolarmente efficace nel fornire risposte accurate e pratiche, arricchite dall'esperienza diretta di utenti che hanno affrontato problemi simili, garantendo così una maggiore pertinenza e utilità.

3.3.3 - 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

manca 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 la 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 iniziale o una comprensione generale del problema, 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

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] A. Aizawa, «45-65,» 2003. [Online].

Available:

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

[5] 9 June 2024. [Online].

Available: <https://www.thesundayprompt.com/autoprompting-facciamoci-aiutare-direttamente-da-chatgpt/>.