Introduzione

Per questo progetto utilizzerò diverse risorse e tecnologie chiave:

- Stack Overflow Stack Overflow è un sito web ampiamente utilizzato dalla community per porre domande su vari argomenti di programmazione e non solo. Quando un utente pubblica una domanda, questa diventa visibile a tutta la community, che si occupa di fornire le risposte. Le risposte possono essere votate: quelle con più voti vengono indicate come "migliori risposte" dalla community, in quanto considerate le più corrette o soddisfacenti.
- Algoritmo TF-IDF L'algoritmo TF-IDF (Term Frequency-Inverse Document Frequency) è una funzione usata per misurare l'importanza di un termine all'interno di un documento o di una collezione di documenti. Questo valore aumenta proporzionalmente al numero di volte che il termine appare nel documento, ma cresce inversamente proporzionale alla frequenza del termine nell'intera collezione di documenti. Per questo progetto, utilizzerò la libreria sklearn.feature_extraction.text, importando il modulo TfidfVectorizer.
- Sistemi di Intelligenza Artificiale (IA) I sistemi di intelligenza artificiale sono progettati per simulare l'intelligenza umana. Questi sistemi utilizzano ampie raccolte di dati seguite da una fase di progettazione e programmazione di modelli predittivi. Nel nostro caso, utilizzeremo un modello di generazione di testo per rispondere alle domande. Useremo ChatGPT, una chatbot sviluppata da OpenAI, basata su modelli di linguaggio avanzati addestrati su un vasto corpus di dati testuali, che permette di generare risposte coerenti e rilevanti in base alle domande poste.

Questo progetto combina le potenzialità delle community online, delle tecniche di elaborazione del linguaggio naturale e dei sistemi di intelligenza artificiale per fornire risposte efficaci e utili alle domande di programmazione.

Primo passo

Nella prima fase progettuale, dovremo prendere il dataset di domande di stackoverflow, ricavato dal sito Archive.org e catalogare il dataset attraverso varie tecnologie, una di queste è l'algoritmo TF-IDF. Per fare ciò prenderemo l'XML dei post di StackOverflow e per ogni PostTypeId=1 ricaveremo le domande. Dopodiché puliamo le domande dai tag HTML tramite "bs4" importando la classe "BeautifulSoup", inoltre importando il modulo "sklearn.feature_extraxtion.text" della libreria "scikit-learn" per utilizzare la classe "TfidfVectorizer" inclusa nel modulo. A TfidfVectorizer passo tre parametri: max_df, min_df e stop_words.

- Max_df rappresenta la frequenza massima dei termini che verranno considerati nel calcolo del TF-IDF. Ad esempio, se si imposta max_df=0.95, il TfidfVectorizer ignorerà i termini che appaiono in più del 95% dei documenti. Questo parametro aiuta a rimuovere i termini molto comuni che non sono rilevanti per la distinzione dei documenti.

- Min_df rappresenta la frequenza minima dei termini che verranno considerati nel calcolo del TF-IDF. Per sempio se si imposta min_df=2, il TfidfVectorizer ignorerà i termini che appaiono in meno di due documenti. Questo aiuta a rimuovere i termini poco frequenti che potrebbero essere rumore nella rappresentazione dei documenti.
- Stop_words parametro che rappresenta un elenco di parole comuni che saranno ignorate durante il calcolo del TF-IDF. Le "stop words" sono parole molto comuni che non aggiungono significato al testo, come "the", "and", "is", ecc. Impostando stop_words='english', ad esempio, si utilizzeranno le stop words predefinite per la lingua inglese fornite dalla libreria “scikit-learn”.

Purtroppo, la libreria fornisce un numero limitato di stopwords. Per ovviare a questo, è stato integrato un vocabolario aggiuntivo a quello di base fornito da 'nltk'. Questa estensione consente di ampliare il dataset di stopwords e, di conseguenza, migliorare l'accuratezza del nostro modello.

Una volta ottenute le informazioni dall'applicazione dell'algoritmo, le inseriamo in un file JSON. Per ogni post, questo file conterrà i punteggi TF-IDF di ogni parola significativa individuata dall'algoritmo. Infine, verrà indicata la parola con il punteggio TF-IDF più alto, che utilizzeremo come termine di catalogazione.

Questo è fondamentale per la catalogazione delle domande e delle risposte. Effettuiamo diverse operazioni di catalogazione con l'obiettivo di sviluppare uno script specifico in base al tipo di domande e risposte per il secondo passo del processo. Pertanto, distinguiamo tra:

- Le domande e le risposte che contengono codice.
- Le domande che hanno un numero di caratteri inferiore e maggiore a 700.
- Le domande che non hanno risposte.
- Le domande che in base al calcolo del TF-IDF hanno delle keywords specifiche.

Secondo Passo

Nella seconda fase del progetto, dovremo gestire le catalogazioni delle domande e delle risposte in modo differenziato. I risultati sono stati salvati in file JSON per consentire interrogazioni tramite la libreria 'openai' e ottenere così informazioni rilevanti per noi.

Per raggiungere questo obiettivo, estraiamo le domande da tutte le categorie di catalogazione e le passiamo a una funzione che si occupa di configurare il JSON di output. Questa funzione, a sua volta, chiamerà un'altra funzione incaricata di inviare le domande a ChatGPT e restituire le risposte. In questo modo, otterremo una risposta dall'IA per ciascuna domanda.

Oltre a queste interrogazioni, eseguiamo un'ulteriore verifica chiedendo a ChatGPT se la migliore risposta trovata su StackOverflow è equivalente alla sua risposta. Questo passaggio ci permette di verificare se ChatGPT comprende correttamente la domanda e fornisce una risposta adeguata ma anche di vedere se le risposte di StackOverlfow sono inerenti alle domande.

Questi sono i passi eseguiti per tutte le tipologie di catalogazione. In aggiunta, per il file che contiene domande e risposte con del codice, verrà effettuata un'ulteriore richiesta a ChatGPT. Questa richiesta consisterà nel rilevare la presenza di codice nelle domande e risposte e verificare se quel codice compila correttamente o, nel caso di frammenti decontestualizzati, se è semanticamente corretto. La risposta a questa interrogazione verrà poi inserita nel file JSON in modo strutturato, per facilitare l'estrazione delle informazioni e consentire una successiva analisi statistica dei risultati.

Per ottimizzare il codice e migliorare le performance quando si gestiscono un gran numero di post, utilizzo la libreria 'concurrent.futures' importando il modulo 'ThreadPoolExecutor'. Questo mi permette di lanciare più thread simultaneamente, accelerando così il tempo di esecuzione delle richieste. Inoltre, utilizzo la libreria 'functools' per implementare una cache che memorizza i risultati delle domande e risposte già analizzate. In questo modo, evito di eseguire nuovamente analisi su dati già trattati e prevenendo richieste duplicate.

Al termine dell'esecuzione del programma, riceveremo tutti i file JSON contenenti le risposte di ChatGPT alle domande oltre alle migliori risposte si Stack Overflow, insieme ai risultati delle interrogazioni effettuate, come l'equivalenza delle risposte e la presenza di codice, inclusa la verifica della sua compilabilità.

Terzo Passo

Nella terza fase del progetto, analizzeremo i file JSON ottenuti dal secondo passo per estrarre e ricavare dati leggibili tramite testo e diagrammi. A tal fine, utilizzerò la libreria matplotlib.pyplot, che sarà impiegata per costruire i diagrammi, in particolare un istogramma, una volta letti i JSON e ricavati i dati fondamentali.

Per le domande che superano o non raggiungono un determinato numero di caratteri limitChar e per quelle che contengono parole chiave specifiche, analizzerò l'equivalenza tra la migliore risposta di Stack Overflow e quella di ChatGPT. Invece, per le domande e risposte contenenti codice, valuterò se il codice presente nelle risposte compila correttamente o meno.

Per le domande con più o meno caratteri rispetto a limitChar e quelle caratterizzate da specifiche parole chiave, raccoglierò il numero di risposte equivalenti e non equivalenti per valutare le capacità di ChatGPT. Per le domande e risposte contenenti codice, verificherò la presenza di codice nelle risposte di ChatGPT e Stack Overflow, restituendo il numero di risposte di ChatGPT che contengono codice e se questo compila, e farò lo stesso per le risposte di Stack Overflow.

Inoltre, presenterò gli stessi risultati in formato testuale, tramite una funzione che, prendendo i dati ottenuti, li scriverà in modo chiaro e leggibile.

Scopo progetto tesi

Da un dataset basato su StackOverflow.com dove sono presenti domande su determinati argomenti e le migliori risposte su di esse . Catalogare le domande e le risposte.

In seguito agganciare il tutto a sistemi IA come chatGpt per effettuare un confronto tra appunto le migliori risposte di StackOverflow e le risposte di chatGpt.