AI Developer Test
NBAgpt

Sanvito Alessio

11/2023

Abstract

L'obiettivo di questo progetto è quello di costruire un chatbot intelligente in grado di interpretare le richieste dell'utente ed estrarre informazioni rilevanti da un dataset tabellare strutturato per fornire risposte precise. In particolare, i dati rigardano l'NBA e comprendono dati sulle partite dal 2004 al 2020, contengono informazioni sia a livello individuale di ogni giocatore che a livello di squadra. Questo chatbot viene creato utilizzando il framework Langchain e l'interfaccia per interagire con esso viene creata con Streamlit.

Keywords: *Large Language Models, NBA, Pandas AI, Langchain, Pandas Dataframe Agent*

---

1. Dataset

Per svolgere questo progetto sono stati scaricati da Kaggle tutti i dataset presenti all'interno del dataset 'NBA games data'; in particolare i dataset sono:

- games.csv: tutte le partite dalla stagione 2004 all'ultimo aggiornamento (dicembre 2020) con la data, le squadre e alcuni dettagli come il numero di punti; comprendono tutte le statistihe di squadra riguardo ogni partita
- games_details.csv: dettagli del dataset delle partite, tutte le statistiche dei giocatori per una determinata partita
- players.csv: dettagli dei giocatori (nome, id, squadra, stagione)
- ranking.csv: classifica NBA di un giorno specifico (divisa in ovest ed est nella colonna conference)
- teams.csv: tutte le informazioni di ogni squadra come città, capacità dell'arena, proprietario, coach ecc.

La prima operazione svolta è stata quella di capire il tipo di dato con cui si sarebbe dovuto lavorare, perciò ogni dataset è stato analizzato per comprendere a fondo a cosa servisse ogni colonna, per capire la struttura dei dataset, per capire come poter eventalmente collegare tra di loro alcuni di questi dataset e quindi avere un'idea precisa su tutti i campi dei vari dataset.

Durante questa fase è subito stato notato che in quasi tutti i csv sono presenti valori NaN che vanno gestiti. Questo è stato fatto nella fase di preprocessing.

2. Preprocessing

Sono stati eseguiti diversi step di preprocessing, partendo dalla rimozione delle istanze (righe) con valori NaN nella maggior parte delle colonne; queste infatti sono state ritenute poco significative in quanto non portano valore aggiunto a un'analisi, ma anzi rischiano solamente di comprometterla. Inoltre, essendo i csv molto ricchi di informazioni (per esempio il dataset contenente le partite dal 2004 al 2020 presenta un numero di righe superiore a 26000) ed essendo le righe che presentano valori NaN ridotte in numero (al massimo qualche centinaio) si è deciso di eliminare queste righe poco informative.

Iniziando con la prima tabella (games.csv) si è notato che il numero di righe che presentavano valori nulli è 99 e in questo caso tutti i valori delle statistiche di quella partita sono NaN. Perciò il dataframe è stato privato di queste. Un discorso simile può essere fatto per games_details.csv dove sono raccolte tutte le statistiche di ogni giocatore in ogni partita (sia che abbia giocato, sia che non abbia giocato); in questo caso si è notato che le righe che presentavano valori nulli sono associate a quei casi in cui il giocatore non è stato convocato. Infatti nella colonna 'COMMENT' per queste righe il commento può essere 'DNP - Coach's Decision' o 'DNP - Injury/Illness'; nonstante possa essere un dato importante per possibili domande sul numero di partite saltate da un giocatore, questo potrebbe essere anche reperito in altro modo, perciò si è deciso di eliminare queste 110000 righe circa e lasciandone circa 558000. Per lo stesso motivo si è deciso di eliminare le righe con valore '0:00' nella colonna 'MIN'.

Il csv ranking presenta valori NaN solo nella colonna 'RETURNTOPLAY' che presumibilmente indica le partite che sono state sospese per la pandemia e che sono state giocate alla ripresa di questo stop; non sono state fatte operazioni su questa colonna.

Infine la tabella che contiene tutte le informazioni riguardanti le squadre contiene dei valori NaN nella colonna relativa alla capacità di spettatori dell'arena. Dal momento che sono pochi valori, sono inseriti a mano cercando di inserire dati il più possibile accurati.

A questo punto, avendo dei dati più puliti, si è deciso di fare ulteriori analisi e step. Nel dataframe games_details è stata aggiunta una colonna che permette di considerare i minuti in maniera da poterli sommare, portando quindi i secondi in formato decimale. Quindi un esempio potrebbe essere un giocatore che ha valore 18:06 per una certa partita nella colonna 'MIN', avrà 18.1 nella nuova colonna. Questo permette di ottenere un'analisi più accurata potendo includere anche i minuti giocati come filtro.

Un'altra operazione è stata quella di trasformare il formato della colonna 'GAME_DATE_EST' in datetime per fare in modo che potesse essere letta correttamente come data.

Un'ulteriore operazione che è stata fatta è stata quella di aggiungere la colonna 'SEASON' all'interno del df 'games_details' in modo tale da poter evitare operazioni inutili di join tra tabelle nel momento in cui vengono chieste informazioni su statistiche di un certo giocatore riguardo a una o più stagioni. Questo è stato fatto tramite una left join dei dataframe 'games_details' e 'games' (di cui sono state considerate solo le colonne 'GAME_ID', 'SEASON') sulla colonna 'GAME_ID'.

Avendo ora dei dati sicuramente più puliti e con informazioni rilevanti, si è passati alla fase di sviluppo del chatbot.

3. Chatbot Developement

Per lo sviluppo e la realizzazione del chatbot è stato utilizzato Langchain, un framework per lo sviluppo di applicazioni alimentate da Large Language Models. In particolare, dovendo trattare dati in formato tabulare, sono stati individuati due tipi di agenti che potrebbero essere utilizzati in grado di fare ciò: "csv agent" e "pandas dataframe agent".

Dopo aver testato le capacità di entrambi e non avendo notato grosse differenze nelle risposte, siccome nella descrizione del csv agent è presente una nota che spiega che quell'agente chiama il modulo 'Pandas DataFrame' (che a sua volta chiama il modulo Python), si è deciso di usare direttamente il pandas dataframe agent.

Per fare ciò è stato seguito il Multi DataFrame Example che permette di passare all'agent

diversi dataframe e farlo interagire con quelli che sono stati passati come una lista. Come modello è stato scelto GPT4 utilizzando il LLM ChatOpenAI che è più adatto e friendly quando lo scopo è costruire un chatbot.

Inizialmente, provando a fare delle domande specifiche l'agent alterna poche risposte giuste a parecchie risposte sbagliate (capita anche che non riesca a rispondere) dettate anche dal fatto che il prompt non è specifico per questa task e per questi dati.

Si è deciso quindi di andare a lavorare sul prompt facendo un lavoro consistente di prompt engineering. In primo luogo è stato definito il ruolo dell'agente, ovvero il fatto che si tratta di una friendly AI incentrata su dati NBA che deve rispondere a domande sull'NBA poste da fans. Successivamente viene fatta una descrizione dei dataframe che vengono passati all'agente, cercando di fare in modo che possa capire in autonomia dove recuperare le informazioni per rispondere alle domande.

Inoltre vengono definiti dei comportamenti di default che l'agent deve rispettare (usare i CSV per rispondere alle domande, se non trova la risposta non deve inventare niente, ma semplicemente dire che non ha trovato la risposta ecc.).

Infine, prima di lasciare il prompt di default richiesto da questo tipo di agent, vengono anche date delle informazioni più dettagliate per definire un comportamento più specifico in alcuni casi che davano ancora alcuni problemi; in particolare è stato definito che, quando si risponde a domande riguardanti la sintesi delle statistiche di carriera di un giocatore, è possibile creare un riassunto effettuando operazioni sulle colonne (come la mediana o la media) o nel caso di domande su un giocatore che chiedono se ha giocato almeno un certo numero di minuti in una o più stagioni, è necessario raggruppare per 'PLAYER_ID' e 'SEASON' e quindi sommare tutti i valori nella colonna 'TOTAL_MINUTES' corrispondenti a quel giocatore in quella/e stagione/i o altre informazioni di questo tipo.

Combinando tutte queste informazioni con il prompt di sistema si ottiene un prompt molto lungo, ma preciso che riesce a guidare il LLM nel ragionamento in maniera corretta.

Si è dunque passati alla fase successiva ovvero quella di creare un'interfaccia tramite cui interagire con il chatbot.

4. Chatbot UI

La web app che è stata implementata chiede in primo luogo di inserire la chiave 'OPENAI_API_KEY' per fare in modo che venga utilizzata quella chiave all'interno delle chiamate al LLM. Una volta inserita la chiave, all'interno dell'applicazione compare la possibilità di fare due azioni tramite un menu a tendina: sfruttare il chatbot per fare domande oppure chiedere di fare qualche grafico per visualizzare alcuni dati contenuti in uno dei csv a disposizione.

4.1. Funzionalità 'Ask your data'

Questa funzionalità è quella che permette di porre domande all'agent che cercherà la risposta all'interno di tutti i dataframe a disposizione in modo tale tale da rispondere il più esaustivamente possibile alla domanda dell'utente.

Questa interfaccia viene implementata come un normale sistema di messaggistica e all'interno di questa conversazione si ha la possibilità di mantenere la memoria della della conversazione; viene implementato anche un bottone per eliminare e resettare la memoria dell'agente.

In Figura 1 si può vedere un'anteprima di questa schermata.

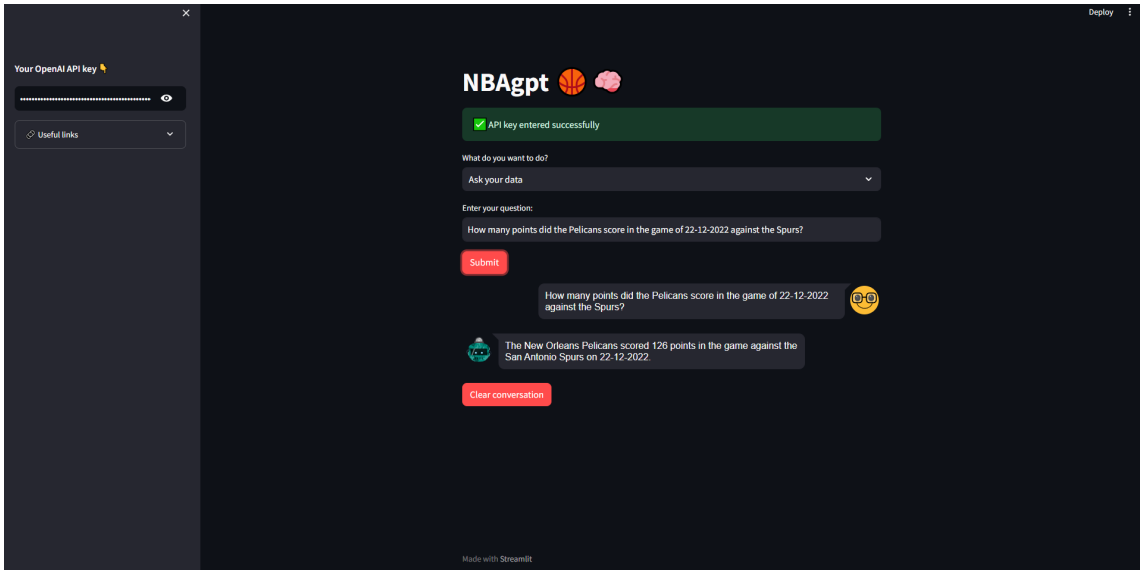

Figure 1. Ask your data

4.2. Funzionalità 'Ask for some plot'

La seconda funzionalità offerta all'interno di questa web app permette di visualizzare i dati di un dataframe (in un pool di 6 dataframe che sono quelli utilizzati nel chatbot). Dunque è possibile selezionare il dataset che piu si preferisce e interrogarlo (questa volta sfruttando la libreria PandasAI) chiedendo di plottare alcuni dati e ottenendo una risposta in forma grafica. Inoltre nella sidebar laterale è possibile vedere una descrizione del dataset, mentre nella schermata principale è possibile vedere un'anteprima del dataset per avere anche un'idea della domanda che potrebbe essere posta.

Nelle figure 2, 3 si possono vedere due screen della web app in questa funzionalità.

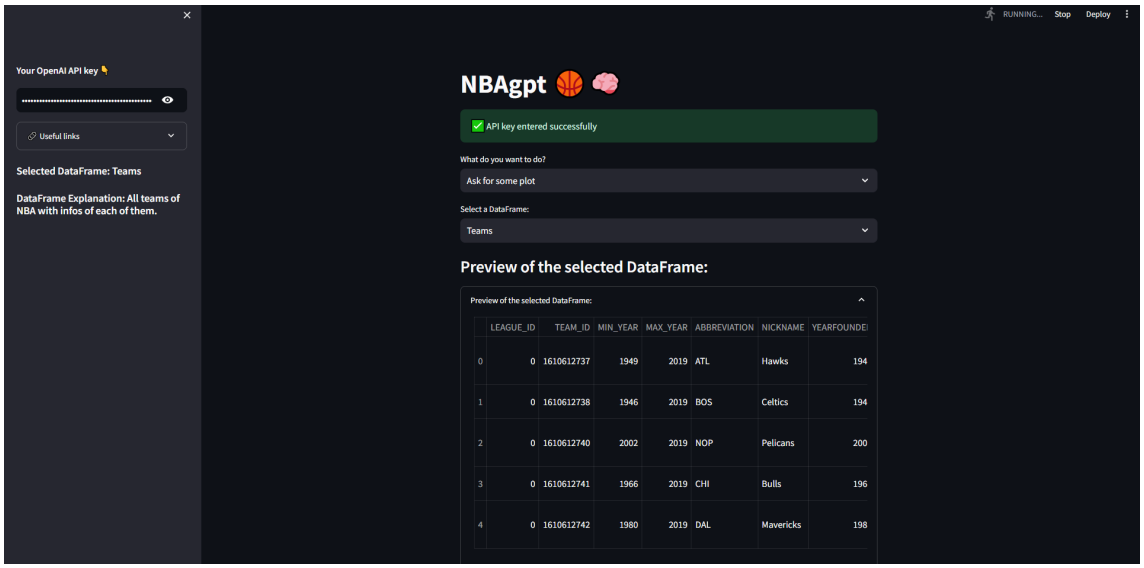

Figure 2. Ask for some plot (parte 1)

Figure 3. Ask for some plot (parte 2)

5. Sviluppi Futuri

Come sviluppi futuri ho pensato che un'idea potrebbe essere di creare un menu a tendina anche per far scegliere all'utente il LLM che vuole utilizzare per rispondere alle sue domande, fare scegliere all'utente quindi se usare ad esempio GPT-4 o GPT-3.5 o un modello open source. Questo permetterebbe anche di valutare le diverse performance dei diversi LLM su questo tipo di task.

Un'ulteriore idea è quella di aggiungere di nuovo le informazioni sugli infortuni o sui giocatori non convocati per scelta del coach (che sono stati rimossi in quanto presentavano solo valori NaN nelle statistiche) attraverso un dataframe dedicato (che non contiene statistiche, ma magari solo informazioni generiche sulla partita), in modo da includere anche i dettagli su quei giocatori durante quelle partite fornendo un'altra dimensione di analisi.

Un altro sviluppo futuro potrebbe essere quello di estendere la parte di visualizzazione. Infatti, tramite pandasAI, è stato complicato ottenere un risultato semplice come quello mostrato; sebbene anche pandasAI dia la possibilità di poter utilizzare diversi dataframe in input (creando uno SmartDataLake), al momento la performance non risulta paragonabile a quella del pandas agent di Langchain. Infatti, il funzionamento di PandasAI è decisamente peggiore e non porta mai a risultati concreti.