Mistral7b Instruct for Tabular Data Generation

Fabrizio Tempo
matr. 247195
tmpfrz00m01a773n@studenti.unical.it

***Abstract*—Effettuare generazione di dati semistrutturati nella forma di tabelle è un task che tradizionalmente si poteva risolvere mediante reti Bayesiane. Tuttavia, recentemente sono emersi nuovi approcci basati su architetture dense come VAE,GAN e LLM che hanno mostrato risultati promettenti. Di contro, generare collezioni di dati semistrutturati mediante Large Language Model è, ancora oggi, un task ampiamente discusso e di difficile risoluzione (X. Fang *et al.*, [1]). In questo progetto ci siamo concentrati sul finetuning di un LLM su questo task.**

***Index terms*—LLM, Table generation, Fine tuning, Prompting, Table serialization**

I. Introduzione

Nonostante le vaste capacità di LLM di apprendere nuovi task, il loro utilizzo per operare su dati semistrutturati nella forma di tabelle è ancora oggetto di studio. La maggior parte degli sforzi e dei successi si sono concentrati sull'apprendimento di task di classificazione e predizione su dati in forma tabellare usando anche architetture transformer come BERT.
Di contro, sul task di generazione di dati semistrutturati realistici in forma tabellare si hanno avuti pochi successi effettivi (V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, [2]) e il motivo è dato dal fatto che i dati semistrutturati in forma tabellare presentano varie caratteristiche che rendono difficile l'apprendimento del processo di generazione.
In particolare, i principali problemi sono rappresentati da:

- presenza di rumore sui dati;
- permutazioni degli attributi e delle righe della tabella;
- eterogenerità dei tipi delle feature dei dati.

Un appoccio tradizionalmente usato per efettuare l'apprendimento di questo task mediante LLM consiste nell'effettuare una trasformazione dei dati semistrutturati in forma tabellare non sequenziale in rappresentazioni testuali sequenziali (V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, [2]) che sono sintatticamente corretti rispetto i valori associati alle features originali e che introducono informazioni aggiuntive che rendono possibile il processo di apprendimento del LLM.
In questo progetto abbiamo però voluto seguire un approccio alternativo: effettuare il finetuning direttamente sui dati in forma tabellare (cioè senza effettuare la trasformazione degli stessi in forma di frasi) andandoli a rappresentare in maniera opportuna.
Rispetto all'utilizzo di LLM per effettuare tabular data generation si è inoltre visto in letteratura come al crescere della complessità e della dimensione del modello si accrescono conseguentemente le performance complessive dello stesso (X. Fang *et al.*, [1]) e per questo motivo la maggiorparte delle ricerche in questa direzione si sono concentrate su modelli GPT (GPT3, GPT3.5 e GPT4). In opposizione a ciò abbiamo voluto valutare la capacità di un modello più piccolo nel risolvere un task di questo tipo.
Infine, esistono diversi dataset attualmente disponibili in forma tabellare (csv o json), ma al crescere dell'eterogeneità delle feature e della loro astrazione si accresce anche la difficoltà interpretativa dei risultati ottenuti, per questo motivo abbiamo selezionato un dataset tale che fosse semplice da interpretare e che avesse poche features.

A. Contenuto del lavoro

I principali sforzi svolti nello svolgimento di questo progetto sono stati:

1. Raccolta di un dataset di buona qualità e di dimensione compatta;
2. Formattazione del dataset in un formato apprendibile dal modello;
3. FineTuning del modello sul task di Tabular Data Generation nella forma di Question Answering;
4. Valutazione dei risultati ottenuti rispetto a una metrica di riferimento;
5. Variazione delle performance rispetto a varie forme di prompting.

Nelle sezioni successive vedremo nel dettaglio come abbiamo realizzato quanto detto e i risultati ottenuti.

II. Model Selection

Il punto di partenza è stata la selezione del modello sul quale effettuare il finetuning. Come evidenziato in Sezione I, i migliori risultati si ottengono usando modelli complessi e di grandi dimensioni come GPT3.
In opposizione a ciò, abbiamo ricercato dei modelli meno complessi e più piccoli, ma tali che avessero delle performance complessive della versione pretrained robuste.

La scelta finale è ricaduta sul modello **Mistral7b** in versione Instruct e ciò ci ha imposto di effettuare il finetuning sul task di tabular data generation come forma di Question Answering.
Questo ha implicato, come vedremo in Sezione III, l'aggiunta di una fase di trasformazione del dataset in un insieme di domande e risposte formattate in accordo alle specifiche del modello.

III. Dataset

La maggior parte dei dataset per machine learning classici consistono in dati in forma tabellare, ma per effettuare una selezione di questi dobbiamo tenere conto di alcune caratteristiche che dovremmo evitare:

1. dati rumorosi;
2. dimensionalità;
3. eterogeneità dei tipi delle features;
4. difficile esplicabilità delle features.

In particolare, i due elementi sui quali abbiamo fatto maggiore attenzione sono stati la dimensionalità dei dati e l'esplicabilità delle features.
Questo è dato dal fatto che nel primo caso vogliamo avere poche features in modo da avere tabelle più compatte, con meno correlazioni tra varie features e soprattutto che richiederanno meno memoria per essere allocate in fase di training; mentre nel secondo caso è per garantire, nella fase di valutazione del modello, la facilità della verifica della correttezza degli output prodotti dal modello.
Per questo fine abbiamo selezionato un campione di dati estratti da **Jefit**.

A. *Jefit Routines Dataset*

Jefit è una piattaforma online che include al suo interno un ampio database di routine di allenamenti per la palestra realizzati sia dai gestori del sito che dagli utenti.
Le varie routine sono indicizzate all'interno di una pagina indice, come in Fig. 1, dove ad ogni routine sono associate delle parole chiave che descrivono la routine insieme al nome e al creatore della stessa.

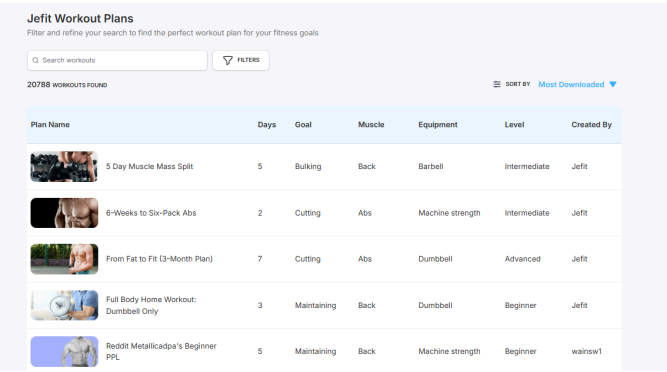

Fig. 1: Pagina indice in cui sono indicizzate le routines.

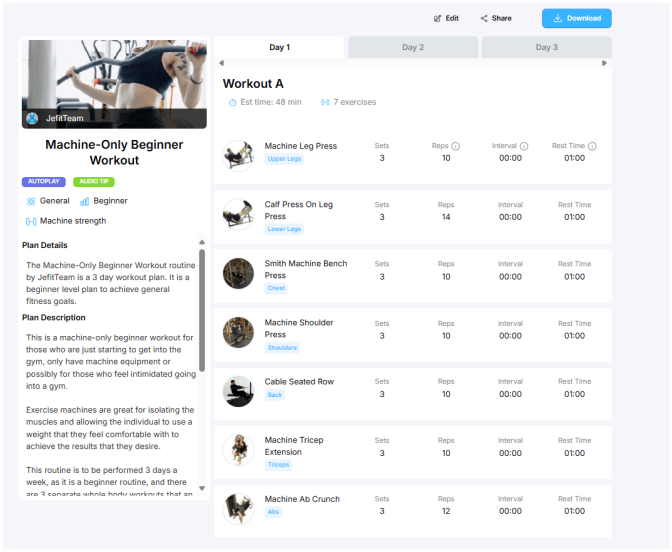

Fig. 2: Pagina associata ad una routine.

Nella Fig. 2 riportiamo invece una pagina associata ad una routine e, in particolare, osserviamo che sulla sinistra è riportato il titolo del piano e una breve descrizione, mentre sulla sinistra i piani di allenamento per ogni giornata suddivisi in diverse tabelle (una per giorno).
Il nostro dataset è stato quindi estratto attraverso l'utilizzo di tecniche di webscraping e si è concentrato nell'estrazione di tutte le routine indicizzate (che avevano un numero di giorni di allenamento in un range fissato) mantenendo per ognuna di esse soltanto le informazioni associate al piano (ovvero le informazioni contenute nell'indice) e le tabelle contenente i piani ad essa associate.
Queste informazioni sono state successivamente aggregate per formare il dataset che utilizzeremo per effettuare il finetuning e la valutazione del modello.

B. *Struttura dataset*

Le informazioni che abbiamo estratto nella fase di webscraping sono state successivamente aggregate all'interno di un singolo file in formato JSON come coppie $(\text{Question}, \text{Answer})$ tali che la Question coincidesse con una stringa composta dalle informazioni descrittive del piano (quelle contenute nell'indice ad esso associato nella pagina indice), mentre l'Answer con una coppia $(\text{Titolo}, \text{Routine})$ e dove il titolo è quello associato al piano.
In particolare, la routine contenuta nell'answer è composta da una lista di tabelle che rappresentano i piani di allenamento per i vari giorni e contenenti gli esercizi che devono essere eseguiti dall'atleta.
Gli esercizi da eseguire in un piano sono caratterizzati da una struttura comune composta da cinque feature principali:

- **Esercizio**: un oggetto caratterizzato dal nome dell'esercizio e dal gruppo muscolare target allenato;

- **Ripetizioni**: numero di volte che si deve eseguire il movimento tecnico;
- **Sets**: numero di volte che si deve eseguire l'esercizio;
- **Rest**: durata della pausa tra un set ed un altro;
- **Interval**: rappresenta la durata temporale per alcuni tipi di esercizi.

La struttura del dataset finale può essere racchiusa dalla seguente struttura:
$[\{\text{Question} : q_i, \text{Answer} : \{\text{Plan Name} : n_i, \text{Routine} : r_i\}\}]$
dove r_i ha la forma
$[[\{\text{Exercise} : e_i, \text{Reps} : r_i, \text{Sets} : s_i, \text{Rest} : r'_i, \text{Interval} : i_i\}]]$
dove e_i consiste in un oggetto descritto da una coppia come segue

$$\{\text{Name} : n_i, \text{Muscle group} : m_i\}$$

Ottengo in fine così un dateset finale composto nel complessivo da **5956** entry totali e che, nella Section IV, vedremo come sarà ulteriormente raffinato per essere utilizzato per addestrare il modello.

IV. Tokenization & Preprocessing

Mistral7b in versione Instruct richiede che il dataset sia opportunamente formattato per poter essere utilizzato come dataset per l'addestramento.
La formattazione del dataset deve essere in modo tale da associare ad ogni coppia nel dataset originale un esempio della forma

$$\text{<s>[INST]Question[/INST]Answer</s>}$$

dove le question sono in questo caso ottenute come combinazione delle question che erano originariamente presenti nel dataset originale con il titoli delle routine contenuto all'interno della answer ad esso associato.
Per quanto riguarda l'answer, questa è stata trasformata tramite tecniche di serializzazione di tabelle in un formato alternativo in modo tale da migliorare le possibilità per il modello di apprendere la struttura della tabella(X. Fang *et al.*, [1]).

A. Serializzazione delle tabelle

La serializzazione delle tabelle è stata effettuata andando a trasformare l'insieme di tabelle associate ad un piano in sequenze di entry con id progressivi nella forma seguente

$$\{0 : \{\text{excercise} : e_0, \text{reps} : r_0, \text{sets} : s_0, \text{rest} : i_0, \text{interval} : i'_0, \text{day} : d_0\}\}$$

dove e_0 consiste solo del nome dell'esercizio tralasciando il riferimento al gruppo muscolare target.
Il dataset così ottenuto viene successivamente tokenizzato usando il tokenizer Mistral di default in modo da ottenere una rappresentazione del dataset utilizzabile per effettuare il finetuning del modello.

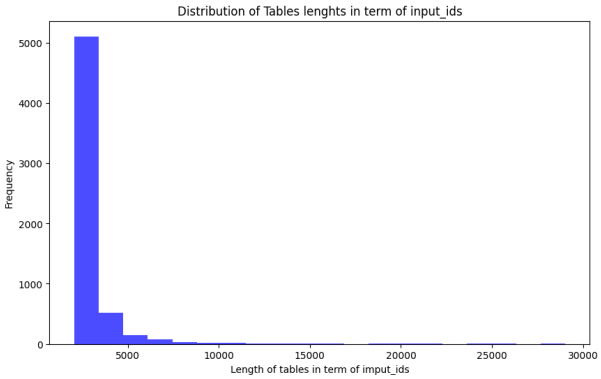

Fig. 3: Frequenze delle lunghezze degli esempi del dataset Jefit.

In particolare, nella Fig. 3 mostriamo come si distribuisce la frequenza degli esempi del dataset rispetto il numero di token. In particolare, osserviamo che la lunghezza media degli esempi è circa 2693 token e come la maggior parte degli esempi hanno una lunghezza inferiore a 5000 token. Per questo motivo se dovessimo effettuare una troncatura raggionevole degli esempi per ridurre le risorse richieste avremmo perdita di informazioni marginali.
Alla fine di questo processo abbiamo ottenuto un dataset che è pronto per essere utilizzato nella fase di finetuning del modello.

V. Training

Ottenuta la rappresentazione del dataset secondo gli standard del modello Mistral7b Instruct abbiamo proceduto con la fase di instruction finetuning rispetto al task di question answering.
Per il finetuning del modello abbiamo sfruttato un ambiente ospitato su Kaggle per cui per limitare le risorse richieste per l'apprendimento abbiamo precedentemente troncato gli esempi del dataset, come visto in Section IV, ma oltre a ciò abbiamo usato altri accorgimenti per ridurre le risorse di calcolo richieste tra cui la Low Rank Adaptation e l'utilizzo della quantizzazione a 4 bit.
Avendo anche limiti sulla durata delle sessioni di addestramento il processo di finetuning è stato suddiviso in step separati eseguiti ciascuno su porzioni diverse del dataset e tali che avessero lo stesso numero di esempi.
In particolare, in ogni sessione si è effettuato il finetuning del modello su una porzione del dataset di dimensione pari al 5% del dataset reale per due epoche considerando come punto di partenza i parametri ottenuti alla fine della sessione precedente. In aggiunta a ciò si è addestrato il modello su circa la metà del dataset in quanto si è visto, in fase di realizzazione del progetto, come proseguendo ulteriormente le performance finali del modello si deterioravano significativamente.
Per quanto riguarda l'ottimizzatore si è deciso di usare AdamW nella versione paged a 8 bit, mentre per gli iperparametri del modello abbiamo usato i valori base per weight decay, dropout e adam ε; per quanto riguarda il learning rate si è deciso di usare

TABLE I: ANDAMENTO DELLA LOSS IN FASE DI TRAINING RISPETTO ALL'ULTIMA SESSIONE DI ADDESTRAMENTO.

Step	Train Loss	Validation Loss
15	0.1831	0.1761
30	0.1603	0.1698
45	0.1610	0.1724
60	0.1548	0.1690
75	0.1474	0.1702
90	0.1486	0.1682
105	0.1504	0.1678

un learning rate decrescente schedulato secondo una funzione coseno.

A. Risultati

Mostriamo nella Table I l'andamento della loss del modello rispetto il training set e il validation set (porzione pari all'1% del dataset e su cui il modello non viene mai addestrato) per cui osserviamo come sia i valori della loss del modello risultano essere sufficientemente bassi per cui possiamo sicuramente scongiurare la possibilità di aver avuto overfitting sui dati di addestramento.
Per carpire la bontà del modello rispetto al task di interesse la sola loss non è sufficente. Per questo motivo nella Section VI vedremo più nel dettaglio quale misura e quali processi abbiamo utilizzato per valutare ulteriormente le performance del modello finale.

VI. EVALUATION

Nel valutare le performance del modello rispetto al task di question answering abbiamo dovuto tenere conto del fatto che le tabelle generate dal modello avranno la stessa struttura di quelle che abbiamo ottenuto in seguito al processo riportato in Section IV.
Per questo motivo osserviamo che nel processo di valutazione del modello abbiamo dato meno peso alla semantica dei termini e maggiore importanza alla forma delle risposte generate.
Per questo motivo un modo in qui la valutazione potrebbe essere effettuata potrebbe consistere con l'andare a confrontare l'output del modello rispetto ad un prompt (ovvero una question del validation set) con la tabella originalmente associata a quel prompt nel dataset formattata secondo la tecnica vista precedentemente.
Osserviamo che l'approccio appena descritto coincide con l'utilizzo della metrica **BLEU** che è proprio quella che è stata utilizzata.

A. Evaluation di base

Per valutare preliminarmente quanto il modello sia migliorato nel risolvere il task rispetto al modello pretrained andiamo a effettuare la valutazione su entrambi considerando una porzione del dataset che non è stata usata in fase di training (pari all' 1% del dataset).
In particolare, calcoliamo la media del BLEU score ottenuta rispetto agli esempi dell'insieme dei dati che abbiamo selezionato come campione per la valutazione.

TABLE II: BLEU SCORE MEDIO SUL MODELLO PRIMA E DOPO DEL FINETUNING.

Model	Mean BLEU
Pretrained	0.0
Finetuned	0.3118

TABLE III: BLEU SCORE MEDIO MODELLO FINETUNED RISPETTO A VARIE FORME DI IN-CONTEXT PROMPTING

Prompting type	Mean BLEU
0-shot	0.3118
1-shot	0.4126
2-shot	0.3578

Nella Table II osserviamo che si è ottenuto un miglioramento significativo delle performance del modello significative rispetto al task in esame.
In letteratura si è osservato anche che l'utilizzo di tecniche di prompting permettono di migliorare le performance generali del modello (X. Fang *et al.*, [1]) per questo motivo procediamo ora nello studiare come variano le performance del modello usando varie forme e combinazioni di prompting.

B. Evaluation prompting base

In letteratura, si è osservato che per il task in esame i principali miglioramenti si hanno effettuando one-shot prompting e two-shot prompting rispetto allo zero-shot prompting con nessun miglioramento sostanziale effettuando #-shot prompting superiore a due e che il modo più semplice per rappresentare i prompt è mediante una breve descrizione del task insieme a degli esempi di tabelle serializzate (X. Fang *et al.*, [1]).
Per questo motivo abbiamo provato inizialmente ad effettuare zero, one e two-shot prompting usando prompt del tipo:

1. nessun esempio, ma solo la domanda per lo zero-shot prompting;
2. "Generate a table that has a structure similar to the following example: {tabella ground truth}. {domanda}" per one-shot prompting;
3. "Generate a table that has a structure similar to the following examples: {prima tabella ground truth},{seconda tabella ground truth}. {domanda}" per two-shot prompting.

Nella Table III mostriamo gli score BLEU per il modello finetuned effettuando zero, one e two-shot prompting e osserviamo come l'introduzione di esempi permetta di aumentar lo score BLEU del modello, sebbene il dal passaggio da one-shot a two-shot prompting non ci permette di ottenere un miglioramento nelle performance. Per questo motivo abbiamo ritenuto il one-

TABLE IV: BLEU SCORE MEDIO MODELLO FINETUNED USANDO ROLE-PLAY PROMPTING CON PROMPT "YOU ARE A PERSONAL TRAINER SPECIALIZED IN TRAINING ATHLETES AND BODYBUILDING, THAT CREATE WORKOUT PLANS BASED ON SPECIFIC REQUESTS.".

	Mean BLEU
Role-play	0.2943

TABLE V: BLEU SCORE MEDIO MODELLO FINETUNED USANDO SELF-AUGMENTED PROMPTING.

	Mean BLEU
self-augmented	0.2528

shot prompting come la versione migliore di prompting tra quelle associate al in-context learning.
Il modo in cui abbiamo effettuato il prompting non è però l'unico possibile e nemmeno il mogliore possibile, per questo motivo proviamo varie tecniche di prompting più avanzate e loro combinazioni.

C. Furter prompt engineering

Un modo che è ampiamente utilizzato in letteratura per effettuare prompting è il **role-play prompting** (X. Fang *et al.*, [1]) che consiste nel fornire nel prompt del modello una descrizione del ruolo che il modello deve ricoprire quando va a costruire una risposta alla richiesta.
Un'altra alternativa è il **self-augmented prompting** (Y. Sui, M. Zhou, M. Zhou, S. Han, and D. Zhang, [3]) che consiste nell'andare a usare la conoscenza a priori di un modello pretrained per effettuare la creazione di un prompt a partire da un esempio.
Ovviamente esistono numerosi modi per creare il prompt in questo caso, noi abbiamo deciso di generare la descrizione del formato a partire da una tabella presente nel dataset originale e combinarla con la richiesta.
In particolare, nella Table IV mostriamo lo score BLEU del modello se si una un prompt role-play, mentre nella Table V quello ottenuto usando la tecnica di self-augmented prompting.
Osserviamo come con il solo role-play prompting riusciamo ad avere performance marginalmente simili a quelle del modello finetuned in cui non si effettua prompting, mentre con il self-augmented prompting non riusciamo a raggiungere nemmeno le performance registrate dal modello finetuned.
Questo potrebbe essere dovuto al fatto che il tipo di informazioni che abbiamo fatto estrarre dal modello pretrained sull'esempio selezionato sono poco significative o non sufficienti per aiutare il modello nella comprensione della struttura delle tabelle.

D. Mixed prompting technique

Vogliamo ora provare a combinare varie forme di prompting per provare ad ottenere degli score migliori rispetto a quelli ottenuti fino ad ora.
In particolare, abbiamo provato le combinazioni seguenti:

TABLE VI: BLEU SCORE MEDIO PER LE VARIE COMBINAZIONI DI PROMPTING CHE ABBIAMO SELEZIONATO.

Combination	**Mean BLEU**
1-shot & role-play	0.3609
self-augment & role-play	0.1267
1-shot & self-augment	0.4740

1. 1-shot prompting & role-play prompting;
2. self-augmented prompting & role-play prompting;
3. self-augmented prompting & 1-shot prompting.

Nella Table VI mostriamo gli score ottenuti dalle varie combinazioni e, in particolare, possiamo osservare come la combinazione one-shot/self-augmented prompting permetta di raggiungere le migliori performance possibili seguito dalla combinazione one-shot/role-play prompting.
Da questo risultato osserviamo come l'aggiunta di un esempio alla descrizione della struttura della tabella, così come l'aggiunta di una breve descrizione del ruolo ad un esempio caratteristico del task ci permetta di migliorare le performance complessive del modello finetuned.

E. Problematiche riscontrate

Nel corso della trattazione abbiamo osservato che che alcuni problemi riscontrati preliminarmente continuano a persistere anche nel modello finetuned e anche in seguito all'utilizzo delle tecniche di prompting.
Questi problemi sono principalmente due:

- **terminazione della tabella generata**: il modello tende a restituire tabelle incomplete dato il prompt;
- **correttezza della tabella generata**: il modello restituisce una tabella nel formato atteso non sentenziale, ma il contenuto delle tabelle potrebbe essere fuorviante o contenere valori (soprattutto se numerici) fuori range o arbitrari.

VII. LAVORI FUTURI

Nel corso della realizzazione diversi elementi sono stati tralasciati o non approfonditi, per questo motivo estensioni di questo lavoro potrebbero consistere in:

1) usare un dataset per il nostro dominio di maggiore qualità;
2) migliorare le performance del modello rispetto la correttezza delle tabelle generate rispetto al dominio di interesse;
3) risolvere il problema sull'incompletezza delle tabelle generate dal modello in fase di inferenza;
4) misurare la variazione di metrica del modello finetuned rispetto a variazioni di prompt per le tecniche di prompting che abbiamo visto;
5) usare tecniche di prompting più complesse;
6) usare prompt tuning.

VIII. Conclusioni

In conclusione, nello svolgimento di questo progetto siamo riusciti a ottenere una versione finetuned di Mistral7b Instruct capace di effettuare generazione di tabelle in forma di question answering usando una forma di serializzazione opportunamente codificata delle tabelle.
In aggiunta a ciò siamo riusciti a migliorare le performance del modello finetuned tramite l'uso di una combinazione di varie forme di prompting cercando, però, di mantenere semplice il prompt somministrato al modello.
Nonostante non si siano risolte le problematiche evidenziate in Section VI.E ci riteniamo comunque soddisfatti dei risultati ottenuti. Infatti, considerando che abbiamo impiegato un modello di dimensioni ridotte e abbiamo usato tecniche di prompting e di serializzazioni semplici siamo riusciti ad ottenere un modello che riesce a generare dati semistrutturati in forma tabellare in forma non sentenziale e tali che siano verosimili.

References

[1] X. Fang *et al.*, "Large Language Models(LLMs) on Tabular Data: Prediction, Generation, and Understanding – A Survey." [Online]. Available: https://arxiv.org/abs/2402.17944

[2] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, "Language Models are Realistic Tabular Data Generators." [Online]. Available: https://arxiv.org/abs/2210.06280

[3] Y. Sui, M. Zhou, M. Zhou, S. Han, and D. Zhang, "Table Meets LLM: Can Large Language Models Understand Structured Table Data? A Benchmark and Empirical Study." [Online]. Available: https://arxiv.org/abs/2305.13062