Elaborazione di tabelle estratte da paper scientifici

[Gabriele Rizzitiello]

Introduzione

Il progetto ha come obiettivo l'automatizzazione dell'analisi di tabelle estratte da articoli scientifici tramite l'utilizzo di Large Language Model (LLM). In particolare, la prima fase del progetto si è basata sulla scelta casuale di 10 paper presenti in un pool iniziale, di circa 300 paper precedentemente estratti da Arxiv. Partendo dai file JSON relativi ai 10 paper scelti, la seconda fase ha riguardato la ricerca di una metodologia, il più vicina possibile, ad un'estrazione automatica dei claim.

Il formato dei claim definitivi richiesto era il seguente:

|{Specification, Specification, ...}, Measure, Outcome|

Specification: **|name, value|**

Struttura dei 10 paper nei file JSON

Ogni file JSON contenuto nella **directory arxiv_10_json** include tutte le tabelle estratte dal relativo paper. I 10 file JSON contengono un totale di 52 tabelle. Ogni tabella nei file JSON possiede i seguenti campi:

1. **Table**: è la struttura della tabella HTML.
2. **Caption**: una breve descrizione del contenuto della tabella, utile per comprendere la natura dei dati.
3. **References**: paragrafi nel paper che fanno riferimento alla tabella stessa, fornendo ulteriore contesto.
4. **Footnotes**: note a piè pagine, poco utilizzate e poco presenti, di conseguenza non sono state prese in considerazione durante le varie fasi del progetto

Tipologie di Tabelle e Ottimizzazione del Codice

Ogni tabella è stata etichettata in base a una delle seguenti **quattro tipologie**:

1. **Relational**: Tabelle con strutture semplici di tipo relazionale.
2. **Nested Relational**: Tabelle che utilizzano strutture relazionali annidate.
3. **Cross-Table**: Tabelle che integrano o comparano i dati tra più tabelle (cross-table).
4. **Nested Cross-Table**: Tabelle che combinano strutture complesse di tipo cross-table con dati annidati.

Manualmente sono state associate tutte le tabelle alla loro tipologia corretta, tramite la costruzione di un dizionario del tipo chiave-valore, dove la chiave è l'id della tabella mentre il valore è il tipo della tabella.

Soluzione iniziale – Full LLM

Un primo approccio possibile è quello di riuscire, disponendo di risorse sufficienti, a scaricare in locale modelli LLM relativi al task di "**Question answering**" o "**Table data interpretation**" necessari per la risoluzione del progetto in questione.

Le problematiche riscontrate durante la messa in atto di questa soluzione sono state per lo più relative alla mancanza di opportune specifiche fisiche delle macchine in nostro possesso. Modelli da noi adottati come ad esempio **FLAN-T5** o **GPT-NeoX-20b** non sono stati all'altezza del compito, in particolare FLAN-T5 (dimensione pesi 6GB) ha una dimensione accessibile ma il task si è rivelato troppo complesso per le sue capacità, mentre GPT-NeoX-20b avendo una dimensione nettamente superiore (dimensione pesi 46GB) anche per task molto semplici avrebbe compromesso il funzionamento dell'intera macchina, saturando l'hardware.

Soluzione definitiva – Hybrid approach

La soluzione definitiva che abbiamo deciso di utilizzare prevedeva un uso combinato di codice e LLM, al fine di semplificare il task generale di elaborazione delle tabelle e favorire un uso più mirato del modello di linguaggio candidato.

Fase 1

Durante questa prima fase lo scopo è stato quello di semplificare un possibile input per il modello LLM, prompt eccessivamente complessi, mal strutturati e prolissi avrebbero deviato il modello dal suo compito. Abbiamo realizzato uno script python chiamato 'JsonProcessor.py' che avvalendosi del dizionario per tipologia di tabella, andasse ad estrarre dalle table i claim in un formato ancora vicino a quello tabellare ma simile al formato desiderato.

La particolarità di questo **claim intermedio**, è che le informazioni estratte sono ancora trattate come se fossero tutte specification, il che rende molto più agevole e comprensibile per il modello LLM l'interpretazione dei dati presenti nella tabella e di conseguenza la produzione dei claim definitivi.

Esempio di claim intermedio ricavato da una tabella relazionale:

['|Method, SimSiam|, |epoch, 400|, ...]

Esempio di claim intermedio ricavato da una tabella nested:

['|Model, CLIP[9]|, |Version, Pre-trained|, |image→text→image text (%),Top-1, 26.680.30|,...]

Model	Version	$image \rightarrow text$ (%)		$text \rightarrow image$ (%)	
		Top-1	Top-2	Top-1	Top-2
CLIP [9]	Pre-trained	$26.68_{0.30}$	$41.80_{0.19}$	$26.17_{0.20}$	$41.13_{0.20}$
PMC-CLIP [43]	Pre-trained	$75.47_{0.37}$	$87.46_{0.11}$	$76.78_{0.11}$	$88.35_{0.19}$
BiomedCLIP [11]	Pre-trained	$81.83_{0.20}$	$92.79_{0.13}$	$81.36_{0.48}$	$92.27_{0.14}$
	InfoNCE [38]	$84.21_{0.35}$	$94.47_{0.19}$	$85.73_{0.19}$	$94.99_{0.16}$
	DCL [39]	$84.44_{0.37}$	$94.68_{0.19}$	$85.89_{0.16}$	$95.09_{0.19}$
	HN-NCE [42]	$84.33_{0.35}$	$94.60_{0.19}$	$85.80_{0.17}$	$95.10_{0.19}$
	DHN-NCE	$\mathbf{84.70}_{0.33}$	$\mathbf{94.73}_{0.16}$	$\mathbf{85.99}_{0.19}$	$\mathbf{95.17}_{0.19}$

TABLE II: Top-K cross-modal retrieval accuracy (mean_{std}) for CLIP models.

A seconda della tipologia di tabella, il metodo relativo produrrà nei claim intermedi coppie e dove necessario anche triple di valori (es. Tabella Nested in figura). I claim intermedi consistono, per tabella, in una lista di coppie e/o triple; in caso di valori vuoti il codice presenterà il termine '–none-'.

Fase 2

Durante la seconda fase lo scopo principale è stato quello di riuscire ad integrare, in modo intelligente e non localmente, l'uso di un modello LLM. É stato necessario realizzare uno scheletro per un prompt univoco che istruisse chiaramente il modello sul come restituire in output i claim nel formato corretto a partire dai claim intermedi in input, in particolare oltre alla lista di claim intermedi, il modello riceveva anche la caption e i paragraph relativi a quella tabella, presentando così anche un contesto.

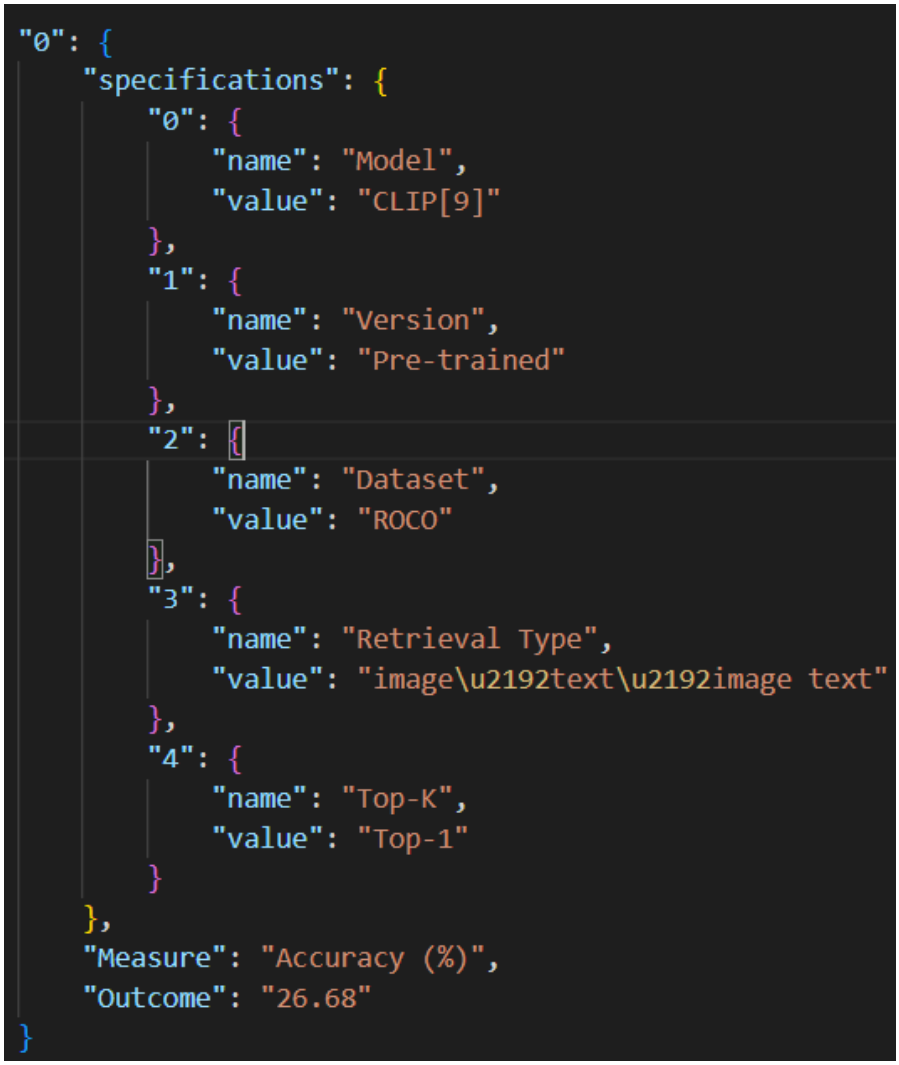
Il **prompt** realizzato è stato il seguente:

```
Ti fornirò:
    -Una lista di celle, organizzate riga per riga, che rappresentano una tabella
HTML. Ogni cella è composta da una coppia {nome, valore}.
    -Una caption della tabella che descrive brevemente il suo contenuto.
    -Alcuni paragraphs come contesto aggiuntivo che possono fornire chiarimenti o
spiegazioni sui dati riportati.

    Il tuo compito è:
    -Identificare e separare le informazioni in specifiche (Specifications) e
metriche di output (Measures).
    -Trattare ogni riga della tabella come una riga di risultato, dove:
    Le specifiche descrivono i dettagli dell'esperimento o del modello (come
dimensione, configurazioni o altre proprietà).
    Le metriche di output descrivono esclusivamente i risultati di valutazione o
performance.
    -Crea un output per ogni riga della tabella nel seguente formato:
    |{|Specification|, |Specification|, …}, Measure, Outcome|

    Dove:
    Specification: una coppia {nome, valore},dove il nome corretto della specifica
(come ad esempio i nomi di colonne) deve essere cercato sia nella table,
    nella caption o nei paragraphs (ad esempio, un modello, un dataset, una
configurazione, un dominio ...),
    mentre il corrispettivo valore dovrà essere cercato nella table.
    Measure: una metrica utilizzata per valutare l'esperimento o il modello, da
ricercare su tutti
    e 3 gli input forniti:lista,caption,paragprah (esempi di metriche sono
"Accuracy", "EER (%)", "F1-measure" ...).
    Outcome: il valore corrispondente alla misura di output (ad esempio, "0.89",
"13.39%" ...).

    Regole:
    -Analizza attentamente i paragraphs e la caption per interpretare quali
informazioni sono specifiche e quali sono metriche di output.
    -Se una riga contiene più metriche di output, suddividi la riga in più righe,
ciascuna con una sola misura e un solo valore di outcome.
```

```
    -Includi eventuali valori mancanti indicandoli come "-none-" dalla sezione
delle specifiche o metriche.

    Esempio generico:

    Dati di input:
    |Model Type, General LLM|, |Model Name, ChatGPT-3.5-turbo|, |Parameter Size,
            175B|, |Level 1, 0.760|
    |Model Type, General LLM|, |Model Name, ChatGPT-3.5-turbo|, |Parameter Size,
            175B|, |Level 2, 0.799|
    |Model Type, General LLM|, |Model Name, ChatGPT-3.5-turbo|, |Parameter Size,
            175B|, |Level 3, 0.408||
Caption: "Table 1: Benchmark results of execution match of all models we tested on
the 'dev' SPIDER dataset."
Paragraphs: "In our experimentation, we organized the models into three distinct
groups as illustred i n Table 1: general purpose LLms,.
    Code-Specific LLMs, and Sequence-to-Sequence models. Table 1 further presents
the Execution Match score on the SPIDER dataset for each studied LLM
    and for each of the four difficulty levels. Note"

    Output atteso:
    |{|Model Type, General LLM|, |Model Name, ChatGPT-3.5-turbo|, |Parameter Size,
      175B|, |Dataset, Spider dev|, |Difficulty Level, 1|}, Execution Match, 0.760|
|{|Model Type, General LLM|, |Model Name, ChatGPT-3.5-turbo|, |Parameter Size,
      175B|, |Dataset, Spider dev|, |Difficulty Level, 2|}, Execution Match, 0.799|
         |{|Model Type, General LLM|, |Model Name, ChatGPT-3.5-turbo|, |Parameter
Size,       175B|, |Dataset, Spider dev|, |Difficulty Level, 3|}, Execution Match,
0.408|

Concentrati solo sui dati forniti e non fare inferenze al di fuori di ciò che è
descritto esplicitamente.
Se qualcosa non è chiaro, interpreta in modo conservativo basandoti su caption e
paragraphs.
Astieniti dal restituire informazioni aggiuntive o note ulteriori che esulano
dall'output richiesto da noi.

    I dati su cui devi lavorare sono:

    -lista:""" + lista + """
    -caption:""" + caption + """
    -paragraphs:""" + paragraphs
```

Il prompt è stato fornito al modello **LLM Gemini1.5** e tramite la relativa interfaccia API è stato possibile automatizzare le richieste (inserendo un opportuno time.sleep() per evitare problematiche di accesso al modello).

- I claim finali ottenuti da Gemini1.5 sono stati parsati ed inseriti nei corrispettivi file.json rinominati con la nomenclatura opportuna: **paperID_tableID_claims.json**.



Creazione spreadsheet – Task 2

Abbiamo analizzato come fossero distribuiti i “nomi” all'interno delle specifiche, per comprendere quali siano i più ricorrenti o significativi. Successivamente, abbiamo esaminato la distribuzione dei “valori” associati a ciascun nome per ogni specifica, così da ottenere una visione più dettagliata delle relazioni e delle variazioni presenti.

Infine, abbiamo approfondito la distribuzione delle “metriche” nel contesto generale, così da identificare eventuali pattern o tendenze rilevanti.

Questo approccio ha permesso di ottenere un profiling completo e ben strutturato delle informazioni estratte. I dati ottenuti sono stati organizzati in 3 coppie di colonne:

	A	B	C	D	E	F	G	H	I	J
1	name	occurrences			value	occurrences			measure	occurrences
2	Method	1,191			SimSiam	1			Top-1 Acc	8
3	Ref	21			[Chen and He,2	1			Accuracy	65
4	Epoch	21			400	3			AVG	10
5	Encoder	21			ResNet-50	8			MemGPU (GB)	4
6	Dataset	1,177			ImageNet-1K	32			Few-Shot Accura	4
7	Base LR	30			MoCo v2	1			Classification Acc	4
8	Warm Up	30			[Chen et al.,202	1			Top-1 Accuracy	13
9	Epochs	15			SimCLR	1			mIoU	203
10	Filters	4			[Chen et al.,202	1			Top 1 Accuracy	6
11	Framework	4			800	3			F1	92
12	Methods	18			BYOL	1			Se	74
13	Backbone	15			[Grill et al.,202(	1			Sp	74
14	Batch Size	12			SwAV	1			Acc	74
15	Data	13			[Caron et al.,20	1			AUROC	73

Alignement – Task 3

Durante il task di alignement lo scopo ultimo è stato quello di allineare sotto lo stesso nome tutti i termini che, concettualmente, sono identificabili con il nome scelto (ad esempio, model e model name sono entrambi identificabili col nome “model name”). Questo obiettivo era richiesto sia per i “name” sia per le “value” delle specifications, sia per le “measures” presenti nei vari claim.

L’approccio è stato il seguente:

- Generazione supervisionata, tramite Gemini1.5, di un dizionario chiave-valore con valore una lista di termini identificabili con lo stesso nome, e come chiave il nome rappresentante

- Uno script python che, sulla base del dizionario, andasse a popolare un file json chiamato RIC_CRI_GAB_ALIGNEMENT.json con un nuovo dizionario chiave-valore con chiave il nome identificativo e come valore una lista di termini assimilati a quel nome, individuati con il formato paperID_tableID_claimID_specificationID

- In RIC_CRI_GAB_ALIGNEMENT.json è presente un dizionario per ogni campo di un claim: ‘name’, ‘value’ e ‘measure’

Casi di particolare interesse

Esistono tuttavia delle rilevanti limitazioni per quanto riguarda LLM in uso, un caso particolare è stato relativo alla tabella **2409.19483_S4.T2.30.30**, per la quale a fronte di una mancata e incompleta interpretazione dei dati ricevuti sono stati ottenuti dei claim definitivi incompleti.

Abbiamo dunque deciso di effettuare un confronto prestazionale anche con **ChatGpt 4** (molto più esteso e non gratuito) che partendo dallo stesso identico prompt ricevuto anche da Gemini1.5 ha mostrato performance nettamente migliori. La differenza tra i due LLM è stata nella capacità di deduzione delle informazioni di contorno offerte da caption e paragraph anche quando questi sono non molto chiari, (si fa riferimento proprio alla tabella in figura mostrata sopra).

Claim definitivo parsato da Gemini1.5:


```
"0": {
    "specifications": {
        "0": {
            "name": "Model",
            "value": "CLIP[9]"
        },
        "1": {
            "name": "Version",
            "value": "Pre-trained"
        }
    },
    "Measure": "Image-to-Text Retrieval Accuracy",
    "Outcome": "26.68"
}
```


Claim definitivo parsato da ChatGpt 4:


```
"0": {
    "specifications": {
        "0": {
            "name": "Model",
            "value": "CLIP[9]"
        },
        "1": {
            "name": "Version",
            "value": "Pre-trained"
        },
        "2": {
            "name": "Dataset",
            "value": "ROCO"
        },
        "3": {
            "name": "Retrieval Type",
            "value": "image\u2192text\u2192image text"
        },
        "4": {
            "name": "Top-K",
            "value": "Top-1"
        }
    },
    "Measure": "Accuracy (%)",
    "Outcome": "26.68"
}
```

Osservazioni e Considerazioni

- Il progetto ha richiesto un **adattamento** del codice per trattare in modo diverso le diverse tipologie di tabella (relational, nested relational, cross-table, nested cross-table).
- Il modello LLM si è dimostrato molto utile per estrarre le specifiche e le metriche dai dati della tabella, una volta che questi erano stati strutturati correttamente.
- Il processo ha reso possibile **semi-automatizzare l'analisi dei dati** nei paper scientifici, risparmiando tempo e migliorando l'efficienza nel trattamento di grandi volumi di dati.

Valutazione della qualità e limiti della soluzione definitiva

La soluzione presentata è particolarmente versatile e adattiva alle varie tabelle incontrate, gestendo in maniera ottimale le elaborazioni sulla base dei requisiti richiesti. Realizzando il ground-truth manualmente e verificando la similarità con i claim da noi prodotti, confrontando claim a claim e specification a specification, abbiamo ottenuto una **similarità totale dell'89%**.

- La precedente valutazione è fortemente dipesa, dalla presenza di tabelle di tipo relazionale, con caption e paragraph che non integrano in modo esaustivo quelle che sono tutte le informazioni necessarie alla creazione di claim ideali.

- La nostra soluzione sarebbe comunque affetta dalla presenza di HTML mal strutturati, poiché durante la prima fase di generazione dei claim intermedi, molto poggia sulla corretta struttura dell'HTML originario relativo alla specifica tabella.

- Un ulteriore limite è l'uso nel dizionario della tipologia delle tabelle, facente uso degli id delle tabelle come chiavi, una soluzione possibile sarebbe l'utilizzo di chiavi univoche. Il dizionario delle tabelle, si basa inoltre su una costruzione manuale (non scalabile), la quale potrebbe comunque essere automatizzata ma sempre tramite l'uso di LLM.

- L'ultimo limite consiste proprio nel tipo di LLM utilizzato: un modello più complesso caratterizzato dalla presenza di un elevato numero di parametri e a pagamento mostrerà sempre delle performance indubbiamente migliori rispetto ad un modello gratuito.

Conclusione

Questo progetto ha combinato l'uso di **analisi semi-automatizzata delle tabelle** con l'applicazione di **modelli LLM**, dimostrando il valore di integrare linguaggi naturali con strutture di dati tabulari per l'interpretazione automatica dei risultati di esperimenti scientifici. Con l'ulteriore ottimizzazione e l'uso su un numero maggiore di articoli, la metodologia potrebbe essere estesa a un'analisi ancora più complessa e sistematica dei risultati scientifici.