Sistema di Data Extraction per articoli su Machine Translation

Alessio Marinucci & Riccardo Felici

https://github.com/DrRicky31/HW4_IID

Indice

1 Introduzione

L'obiettivo di questo progetto sarà estrarre informazioni strutturate da un insieme di file JSON contenenti tabelle presenti in articoli scientifici. Le informazioni estratte includono il nome della metrica, il suo valore e tutte le specifiche correlate (nome e valore). Queste informazioni saranno estratte direttamente dalle tabelle HTML e arricchite con dati contestuali come riferimenti, didascalie e note a piè di pagina per fornire un quadro più completo e accurato.
Il processo di estrazione non si limita a una semplice raccolta di dati, ma utilizza tecniche avanzate di elaborazione del linguaggio naturale (NLP) per identificare con precisione i dati significativi e organizzarli in un formato strutturato. Ciò consente di gestire in modo efficace la complessità e la varietà delle tabelle presenti negli articoli scientifici, che possono includere dati eterogenei e strutture complesse.
Il progetto prevede tre fasi principali:

- **Estrazione dei claims**: Durante questa fase, i dati grezzi vengono analizzati ed elaborati per identificare metriche, valori e specifiche pertinenti. Le informazioni estratte sono strutturate sotto forma di affermazioni leggibili e utili per ulteriori analisi.

- **Profiling**: Successivamente, i dati strutturati vengono sottoposti a un'analisi statistica per calcolare distribuzioni, medie e altre statistiche descrittive. Questa fase consente di comprendere meglio le caratteristiche dei dati estratti e individuare eventuali trend o anomalie.

- **Alignment**: Infine, viene effettuato un allineamento terminologico per standardizzare e unificare i termini utilizzati nelle tabelle. Questo processo coinvolge la creazione di dizionari di sinonimi e l'associazione di termini simili a una forma canonica, migliorando così la coerenza e l'usabilità dei dati.

Il progetto si distingue per l'uso di tecniche innovative basate su modelli di linguaggio (LLM, Large Language Models) sia locali che cloud. In particolare, vengono utilizzati modelli avanzati come **Gemini 1.5** per garantire una maggiore precisione e scalabilità. Questi strumenti non solo migliorano l'accuratezza dell'estrazione, ma permettono anche di adattarsi a diverse configurazioni di dati e contesti applicativi.
Un altro aspetto significativo del progetto è la sua applicabilità pratica. I dati estratti e analizzati possono essere utilizzati in vari settori, tra cui la ricerca scientifica, l'analisi delle prestazioni dei modelli di apprendimento automatico e la sintesi di report automatici. Inoltre, l'approccio modulare adottato consente di estendere facilmente il sistema per includere ulteriori fonti di dati o integrare nuove funzionalità.

2 Struttura della directory

Il progetto è organizzato in una struttura ben definita che suddivide le funzionalità in directory e file specifici. Ecco una descrizione dettagliata:

- **Directory principali:**
  - `alignment`: Contiene gli script e i dati relativi al Task 3 (Allineamento). Le funzionalità principali includono:
    - `alignment.py`: Script principale per l'allineamento delle metriche e delle specifiche.
    - `dict_distribution.py`: Script per aggiornare le distribuzioni calcolate nel Task 2 utilizzando i dati allineati.
    - `merge_alignment.py`: Combina i valori allineati basandosi sui sinonimi, unificando le metriche e le specifiche.
    - `synonym_dict_generator.py`: Genera un dizionario di sinonimi utilizzando modelli di linguaggio, utile per standardizzare la terminologia.
  - `distribution`: Include gli script per il Task 2 (Profilazione). Le funzionalità principali comprendono:
    - `dict_generator.py`: Crea dizionari JSON che mappano metriche e specifiche ai loro valori estratti.
    - `profiling.py`: Calcola distribuzioni statistiche basate sui dati estratti, come distribuzione di metriche e specifiche.
  - `sources`: Contiene i dati di input e script di supporto per il Task 1 (Estrazione di claim). È organizzata come segue:
    - `json`: Sottodirectory contenente i file JSON generati come parte del processo di estrazione delle informazioni.
    - `pdf`: Sottodirectory contenente i file PDF degli articoli scientifici.
    - `classifier.py`: Script per classificare le tabelle e assegnare loro un valore predefinito, utilizzato per elaborare i dati in modo appropriato.
- **File principali:**
  - `claim_extractor.py`: Script principale per l'estrazione di claim strutturati dai file JSON.
  - `classification_mapping.json`: File JSON per mappare le tabelle a una classificazione specifica.
  - `format_json.py`: Script per riformattare i claim estratti in un formato leggibile e standardizzato.
  - `similarity.py`: Script per calcolare la F1-measure dei dati estratti rispetto ad una ground truth.
  - `task1.sh`, `task2.sh`, `task3.sh`: Script di shell per eseguire rispettivamente i Task 1 (Estrazione), Task 2 (Profiling) e Task 3 (Alignment).

3 Task 1: Estrazione delle informazioni

Questo task si concentra sull'estrazione di informazioni strutturate da tabelle presenti negli articoli scientifici. Le affermazioni includono metriche, i loro valori e specifiche associate derivate dalle tabelle e dai loro contesti (es. didascalie, riferimenti e note a piè di pagina). Vengono impiegate tecniche avanzate di elaborazione del linguaggio naturale (NLP), inclusi modelli locali e basati su cloud, per ottenere risultati precisi e affidabili. In quest task sono stati scelti 13 papers di **"Machine Translation"** con un totale di 37 tabelle. Di seguito vengono introdotti i vari scripts che abbiamo utilizzato per svolgere questo primo task.

3.1 Componenti

- `./testing`: Contiene script di test per convalidare singoli tipi di tabelle. Gli output sono memorizzati in `./testing/output_test`.
- `./testing/LLM_testing`:
  - Contiene codice per test che chiama le API dei modelli di linguaggio (LLM) implementati.
  - Utilizza le librerie `Transformers` e `google.generativeai` per LLM locali (basati su BERT) e cloud (Gemini 1.5).
  - L'uso degli LLM locali è stato sostituito con Gemini 1.5 per migliorare le prestazioni.
- Configurazione: Creare un file `./config.py` per configurare la chiave API di Gemini:

  ```
  API_KEY = "YOUR_API_KEY"
  ```

- `./claim_extractor.py`: Elabora tutti i file JSON di input ed esegue le funzioni appropriate in base alla classificazione delle tabelle. Il codice processa le tabelle in input lanciando una funzione specifica per il tipo di tabella, il tipo di tabella viene estratto dal file di classificazione. Dopo aver effettuato una prima estrazione delle informazioni dal formato HTML, utilizzando la libreria `BeautifulSoup`, si integrano i dati con informazioni estratte dalla caption e dai paragrafi che contengono delle riferimenti alla tabella. Tali informazioni sono estratte usando Large Language Model. Inizialmente sono stati implementati modelli locali basati su BERT, specializzati sul question-answering, successivamente sostituiti dal modello in cloud Gemini 1.5, permettendo di avere dei tempi di processamento inferiori e delle performance migliori. È presente un limite di richieste che possono essere fatte al secondo al modello Gemini con il piano gratuito, vengono quindi introdotte delle attese per non superare tale limite. Le informazioni estratte dal modello sono legate principalmente all'estrazione e distinzione delle metriche dalle specifiche e permettono di arricchire le informazioni presenti esclusivamente nel

testo, come ad esempio il nome di una metrica rappresentata in una tabella. Gli output sono memorizzati in `./JSON_CLAIMS` nel seguente formato:

```
[
    {
      "Claim 0": "|{|Dataset, WMT-16|,|Lang. Pair, DE-EN|},
                  # Trn.+Vld., 4,551,054|"
    },
    {
      "Claim 1": "|{|Dataset, WMT-16|,|Lang. Pair, DE-EN|},
                  # Test, 2,999|"
    }
]
```

– `./format_json.py`: Converte le affermazioni JSON in un formato più leggibile. Gli output sono memorizzati in `./JSON_CLAIMS_CONVERTED`:

```
[
    {
        "0": {
            "specifications": {
                "0": {"name": "Dataset", "value": "WMT-16"},
                "1": {"name": "Lang. Pair", "value": "DE-EN"}
            },
            "Measure": "# Trn.+Vld.",
            "Outcome": "4,551,054"
        }
    },
    {
        "1": {
            "specifications": {
                "0": {"name": "Dataset", "value": "WMT-16"},
                "1": {"name": "Lang. Pair", "value": "DE-EN"}
            },
            "Measure": "# Test",
            "Outcome": "2,999"
        }
    }
]
```

3.2 Creazione di una groundtruth con metriche di similarità

L'obiettivo principale è stato quello di verificare se l'estrazione svolta con il sistema, andasse ad individuare la maggior parte se non tutti i claims contenuti nelle tabelle. E' stata svolta un'analisi a mano, considerando tutte le tabelle e definendo delle metriche

per valutare la similarità. Come si vede nel file `similarity.py`, sono state utilizzate le seguenti metriche:

- **Precisione (Precision)**: La proporzione di claim estratti che sono correttamente presenti nei dati di riferimento (ground truth).
- **Recall**: La proporzione di claim di riferimento che sono stati correttamente estratti.
- **F1-Score**: La media armonica di precisione e recall, che fornisce una misura complessiva della performance, soprattutto quando si hanno squilibri tra precisione e recall.

La Ground Truth così ottenuta ha assicurato una base solida per le fasi successive del progetto, garantendo che tutte le valutazioni e comparazioni future potessero essere eseguite con la massima precisione.

3.3 Esempio di Output

Nella seguente sezione vengono riportati i risultati ottenuti dal codice di estrazione dei claim e la relativa ground truth.


```
{
        "4": {
            "specifications": {
                "0": {"name": "Model", "value": "GPT2-small"},
                "1": {"name": "wait-k", "value": "K=9"
                }
            },
            "Measure": "delay improvement performance",
            "Outcome": "0.166"
        }
    },
    {
        "5": {
            "specifications": {
                "0": {"name": "Model", "value": "share-encoder"
                },
                "1": {"name": "wait-k", "value": "K=1"
                }
            },
            "Measure": "delay improvement performance",
            "Outcome": "0.210"
        }
    }
```


La ground truth del Claim 5 risulta leggermente diverso, in quanto la tabella HTML divide il nome del modello in due celle, pertanto non viene correttamente estratto.


```
    {
        "5": {
            "specifications": {
                "0": {"name": "Model", "value": "GPT-small share-encoder"
                },
                "1": {"name": "wait-k", "value": "K=1"
                }
            },
            "Measure": "delay improvement performance",
            "Outcome": "0.210"
        }
    }
```

4 Task 2: Profiling

Questo task analizza le affermazioni estratte per generare dati di profilazione completi. Fornisce approfondimenti statistici calcolando distribuzioni e medie. I risultati sono memorizzati in formati strutturati per una facile interpretazione e analisi ulteriore.

4.1 Componenti

La profilazione include:

- Distribuzione delle metriche.
- Distribuzione dei nomi delle specifiche.
- Distribuzione dei valori per ogni specifica.
- Valori medi associati a ogni metrica.

Gli output sono salvati in `./NAME_PROFILING.csv`.

- `./distribution/dict_generator.py`: Produce dizionari JSON:
  - `./distribution/metrics.json`: Mappa le metriche ai rispettivi valori.
  - `./distribution/specifications.json`: Mappa le specifiche ai rispettivi valori.
- `./distribution/profiling.py`: Calcola distribuzioni dai dizionari estratti.

4.2 Profiling per File e per Paper

Il codice implementa un processo di *profiling* per analizzare e raccogliere informazioni sulle distribuzioni delle specifiche e delle metriche in due set di dati JSON separati, sia nei loro formati originali che allineati. Il *profiling* si concentra su vari aspetti, tra cui la distribuzione delle specifiche, dei valori e delle metriche, nonché il calcolo delle medie per ciascuna metrica. I risultati vengono poi esportati in file CSV per un'analisi successiva.

4.3 Funzioni principali

- **Distribuzione delle Specifiche**: La funzione `spec_distribution()` esamina il primo elemento di ciascun oggetto all'interno del file JSON delle specifiche e calcola la *frequenza di ciascuna specifica*. I risultati vengono salvati in un file CSV con due colonne: 'Specification' (Specifiche) e 'Frequency' (Frequenza).
- **Distribuzione dei Valori**: La funzione `values_distribution()` analizza i valori associati a ciascuna specifica, estraendo il primo valore di ciascun oggetto nel file delle specifiche e calcolandone la *frequenza di occorrenza*. I risultati sono anch'essi salvati in un CSV con le colonne 'Value' (Valore) e 'Frequency' (Frequenza).

- **Distribuzione delle Metriche**: La funzione `metric_distribution()` calcola la *frequenza delle metriche* nel file JSON delle metriche, esaminando i primi elementi di ciascun oggetto. I risultati vengono salvati in un file CSV con le colonne 'Metric' (Metrica) e 'Frequency' (Frequenza).
- **Calcolo delle Medie delle Metriche**: La funzione `metric_averages()` raccoglie tutti i valori numerici associati a ciascuna metrica nel file delle metriche, calcolando la *media* per ogni metrica. Questo passaggio è utile per avere una panoramica delle metriche in termini di valori medi. I risultati vengono salvati in un file CSV con le colonne 'Metric' (Metrica) e 'Average' (Media).
- **Creazione dei File di Output**: I risultati delle distribuzioni e delle medie vengono salvati in due file CSV distinti:
  * `NAME_PROFILING.csv`: Contiene le distribuzioni e le medie per i dati originali.
  * `NAME_PROFILING_ALIGNED.csv`: Contiene le distribuzioni e le medie per i dati allineati. Fa uso di dizionari allineati, descritti nel task 3.

  Ogni file CSV viene aggiornato tramite una modalità di scrittura in append, permettendo di aggiungere nuovi dati senza sovrascrivere quelli già presenti.

4.4 Flusso di lavoro

Vengono prima calcolate le distribuzioni delle specifiche, delle metriche e dei valori per i file JSON originali. Successivamente, viene eseguito lo stesso processo sui file JSON allineati. Il risultato finale è una serie di file CSV che possono essere utilizzati per analizzare la frequenza e la media delle metriche e delle specifiche, fornendo una base per ulteriori analisi.

- **File CSV di distribuzione**: Ogni file contiene la frequenza di specifiche, valori e metriche, sia per i dati originali che per quelli allineati.
- **File CSV di medie**: Ogni file contiene le medie delle metriche per i dati originali e allineati.

Questo approccio consente di avere una visione globale e dettagliata delle caratteristiche dei dati, utile per ulteriori analisi statistiche e di profiling.

4.5 Esempio di Output

Di seguito si mostra un esempio di output del file `./NAME_PROFILING.csv` per vedere il funzionamento del processo di profilazione.

```
Specification,Frequency
Data set,4
MOS,4
Method,13
Output,13
Model,173

Metric,Frequency
Segment Size,4
Pitch,2
Pitch and Energy,2
Output,13
different language model sizes,25

Value,Frequency
Train,1
Development,1
Test,1
Total,1
Without Stress,2
With Stress,2

Metric,Average
Pitch,4.1
Pitch and Energy,4.285
different language model sizes,0.18544
characteristics of test data,315.53125
st-MOS,3.91
Cases,860.35
```

Successivamente invece vediamo una tipologia di output del file `./NAME_PROILING_ALIGNED.csv` per vedere il funzionamento del processo di profilazione una volta fatto l'allineamento dei termini simili.

```
Specification,Frequency
data set,87
mos,6
method,112
"output, paragraphes 1 à 5",13
models,249

Metric,Frequency
Segment Size,4
Pitch,2
Pitch and Energy,2
mos,2
cases,20
frequency,20
1st procedure,20
the delay improvement performance,10
mT5,6

Value,Frequency
Train,1
Development,1
Test,1
Total,1
Without Stress,2
With Stress,2
Original French,1

Metric,Average
Pitch,4.1
Pitch and Energy,4.285
different language model sizes,0.18544
characteristics of test data,315.53125
mos,3.91
cases,860.35
```

5 Task 3: Alignment

Questo task standardizza e unifica la terminologia nelle affermazioni estratte, garantendo coerenza tra metriche e specifiche. I sinonimi vengono identificati, i termini allineati a forme canoniche e le distribuzioni ricalcolate in base ai dati allineati. Il risultato è un dataset unificato pronto per analisi o reportistica.

5.1 Componenti

- `./alignment/alignment.py`: Allinea i valori delle metriche e delle specifiche associandoli a identificatori univoci (`paperID_tableID_claimID_specification_ID`). Gli output sono salvati in `./alignment/aligned_output.json`.
- `./alignment/synonym_dict_generator.py`: Costruisce un dizionario di sinonimi utilizzando un LLM (`All-Mini`). Gli output sono salvati in `./alignment/synonym_dict.json`:

```
{
    "model": ["model", "models"],
    "wait-k": ["wait-k"],
    "dataset": ["data set", "dataset"]
}
```

- `./alignment/merge_alignment.py`: Unisce i valori allineati utilizzando il dizionario di sinonimi. Gli output sono salvati in `./alignment/merged_fields_output.json`.
- `./alignment/dict_distribution.py`: Aggiorna le distribuzioni del Task 2 utilizzando i dati allineati. Gli output sono salvati in `./alignment/NAME_PROFILING_ALIGNED.csv`.

5.2 Creazione del dizionario dei sinonimi

Il codice descritto in questa sezione si occupa della creazione di un dizionario di sinonimi, utilizzando una tecnica di *embedding* delle frasi e una misura di similarità per raggruppare termini simili. Il processo coinvolge vari passaggi, tra cui la normalizzazione dei termini, la generazione di embeddings, il calcolo della similarità tra i termini, la clusterizzazione dei sinonimi e la gestione dei duplicati. Per raggiungere questo obiettivo, utilizza un modello pre-addestrato di tipo SentenceTransformer, nello specifico **all-MiniLM-L6-v2**, noto per la sua capacità di rappresentare frasi e parole in spazi vettoriali semantici. Questo modello permette di calcolare le similitudini tra termini tramite la metrica del coseno tra i loro embeddings.

5.3 Descrizione del processo

Il flusso del processo di creazione del dizionario dei sinonimi può essere suddiviso nelle seguenti fasi principali:

1. **Normalizzazione dei termini**: Prima di eseguire qualsiasi operazione, i termini vengono normalizzati per evitare che varianti di un termine (ad esempio, "data set" e "dataset") vengano trattati come distinti. Questa operazione include la rimozione di spazi extra e la conversione dei termini in minuscolo.
2. **Calcolo degli embeddings**: Successivamente, viene utilizzato un modello di *SentenceTransformer* (in questo caso `all-MiniLM-L6-v2`, ma se ne potrebbero usare altri modelli più specifici per sinonimi, come `paraphrase-MiniLM-L6-v2`) per generare gli embeddings dei termini. Gli embeddings sono rappresentazioni numeriche dei termini in uno spazio vettoriale, che catturano le loro relazioni semantiche.
3. **Calcolo della matrice di similarità**: Utilizzando gli embeddings calcolati, viene creata una matrice di similarità tra tutti i termini, calcolando la similarità coseno tra ciascun paio di termini. Questo approccio permette di capire quanto due termini siano simili dal punto di vista semantico.
4. **Clusterizzazione dei sinonimi**: I termini vengono quindi raggruppati in *cluster* sulla base della loro similarità. Se la similarità tra due termini è maggiore di una certa soglia (ad esempio, 0.45), questi termini vengono raggruppati insieme. Ogni cluster rappresenta un gruppo di termini sinonimi, con uno dei termini scelto come rappresentante del gruppo.
5. **Gestione dei duplicati e creazione del dizionario finale**: Dopo la clusterizzazione, vengono eliminati i duplicati all'interno dei cluster e i termini vengono ordinati per frequenza. Il termine più frequente all'interno di un cluster viene scelto come rappresentante, e tutti gli altri termini del cluster vengono associati ad esso. Vengono poi gestiti i duplicati tra i cluster, assicurando che ogni termine venga assegnato solo a un gruppo.
6. **Salvataggio del dizionario di sinonimi**: Il dizionario finale dei sinonimi viene poi salvato in un file JSON. Ogni chiave nel dizionario è un termine, mentre i valori sono i sinonimi di quel termine, ovvero i termini che sono stati raggruppati insieme.

5.4 Output del Dizionario dei Sinonimi

Il risultato del processo di creazione del dizionario dei sinonimi è un file `synonim_dict.json` in cui ciascun termine è associato a un gruppo di sinonimi. Ogni gruppo rappresenta una collezione di termini che, grazie alla similarità semantica, sono considerati sinonimi tra di loro. Questo dizionario può essere utilizzato in vari contesti, come ad esempio per migliorare la ricerca semantica o l'analisi dei dati, permettendo di trattare diversi termini con significati simili come equivalenti.

```
"dataset": [
    "data set",
    "dataset",
    "characteristics of training data"
],
"mos": [
    "mos",
    "st-mos"
],
"method": [
    "method",
    "iteration method",
    "methods"
]
```

5.5 Processo di Alignment

Il processo di *alignment* descrive come vengono unificati i campi simili all'interno di un dataset, utilizzando un dizionario di sinonimi per raggruppare termini varianti e normalizzare i campi. In particolare, il codice sottostante si occupa di caricare un file JSON contenente i campi e i valori allineati, e di unificarli in un'unica rappresentazione, prendendo in considerazione anche i sinonimi. Questo processo è utile per migliorare la consistenza dei dati e facilitare l'analisi.

Il processo si svolge attraverso diverse fasi principali:

1. **Normalizzazione dei nomi**: Ogni campo nel dataset viene normalizzato rimuovendo spazi, caratteri di underscore e plurali semplici (ad esempio, "field" e "fields" vengono trattati come equivalenti). Questo passo permette di uniformare varianti tipografiche dei termini.
2. **Caricamento del dizionario dei sinonimi**: Viene caricato il dizionario di sinonimi, che associa ad ogni termine un gruppo di sinonimi che possono essere utilizzati in modo intercambiabile. Il dizionario viene caricato dal file JSON che contiene le associazioni tra i termini, visto in precedenza.

3. **Identificazione dei sinonimi per ogni campo**: Per ogni campo del dataset, vengono trovati i sinonimi utilizzando il dizionario. Se un termine è presente in più sinonimi, tutti i sinonimi vengono raggruppati insieme per formare un gruppo di termini equivalenti. Il termine alfabeticamente più piccolo di ogni gruppo di sinonimi viene scelto come rappresentante.
4. **Unificazione dei campi simili**: Una volta identificati i sinonimi, i campi simili vengono unificati sotto un rappresentante comune. I campi che appartengono allo stesso gruppo di sinonimi vengono combinati, ed ogni campo viene associato al suo rappresentante.
5. **Creazione di un nuovo dataset unificato**: Un nuovo dataset viene creato, in cui i campi sono stati unificati. Questo dataset contiene due sezioni principali: `merged_aligned_names`, che mappa i campi unificati alle rispettive identità, e `merged_aligned_values`, che mappa i valori dei campi unificati.
6. **Salvataggio del file risultante**: Il file finale viene scritto in `aligned_output.json`, con i campi unificati e i valori corrispondenti. Questo file rappresenta una versione ottimizzata e normalizzata del dataset originale, utile per l'analisi o per altre operazioni di elaborazione.

5.6 Risultato Finale

Il risultato del processo di alignment è un file JSON in cui i campi simili sono stati unificati in un'unica rappresentazione, riducendo la variabilità nei dati. Le due principali sezioni del file risultante sono:

- `merged_aligned_names`: Mappa i campi unificati alle rispettive identità (spesso rappresentate da ID univoci).
- `merged_aligned_values`: Mappa i valori dei campi unificati, associando ogni valore ai suoi campi unificati corrispondenti.

Questo processo consente di ottenere una versione più coerente e facilmente analizzabile dei dati, utile per operazioni successive di elaborazione e analisi.

5.7 Esempio di Output

In questa sezione viene mostrato un esempio di file di output salvato in `aligned_output.json`, che riflette il processo di allineamento svolto dal sistema, utilizzando il dizionario dei sinonimi visto in precedenza.


```
"aligned_names": {
    "dataset": [
        "2311.14465_3_claims_38_1",
        "2311.14465_3_claims_29_1",
        "2410.07830_1_claims_8_1",
        "2311.14465_3_claims_32_1",
        "2409.17939_2_claims_1_1"
    ],
    "mos": [
        "2403.04178_1_claims_0_1",
        "2403.04178_1_claims_1_1",
        "2403.04178_1_claims_2_1",
        "2403.04178_1_claims_3_1"
    ],
    "method": [
        "2409.17939_1_claims_12_1",
        "2410.06338_3_claims_6_1",
        "2409.17939_1_claims_9_1",
        "2410.06338_4_claims_1_1",
        "2409.17939_1_claims_4_1",
        "2311.14465_3_claims_31_3"
    ],
    "output": [
        "2409.17939_1_claims_0_2",
        "2409.17939_1_claims_1_1",
        "2409.17939_1_claims_4_2",
        "2409.17939_1_claims_5_1",
        "2409.17939_1_claims_6_1",
        "2409.17939_1_claims_7_1",
        "2409.17939_1_claims_8_2",
        "2409.17939_1_claims_9_2",
        "2409.17939_1_claims_10_2",
        "2409.17939_1_claims_11_2",
        "2409.17939_1_claims_12_2"
    ],
```

6 Conclusioni

In questo progetto, è stato sviluppato un sistema innovativo per l'estrazione automatica di informazioni da file JSON contenenti tabelle provenienti da articoli scientifici. L'obiettivo principale era quello di elaborare grandi quantità di dati tabulari in modo strutturato, consentendo di trasformare contenuti non standardizzati in una base informativa adatta all'analisi statistica e semantica.

Il sistema è stato suddiviso in tre task principali:

1. **Estrazione delle informazioni**: Un modulo dedicato all'identificazione delle metriche, dei valori e delle specifiche associate presenti nei dati tabulari. Questa componente ha permesso di arricchire le informazioni estratte con contesti fondamentali, come didascalie, riferimenti e note a piè di pagina.
2. **Profiling**: Un'analisi strutturata delle informazioni estratte, con la generazione di profili statistici completi. Questo task ha consentito di ottenere approfondimenti quantitativi sulla distribuzione delle metriche e delle specifiche presenti negli articoli scientifici, fornendo una base solida per identificare pattern ricorrenti e anomalie.
3. **Alignment**: La standardizzazione della terminologia ha permesso di gestire sinonimi e variazioni linguistiche, garantendo coerenza e uniformità nelle analisi successive. Attraverso l'utilizzo di modelli linguistici avanzati, è stato possibile identificare corrispondenze semantiche, migliorando la qualità e l'affidabilità dei dati.

L'approccio implementato ha raggiunto diversi obiettivi chiave:

- **Efficacia nell'estrazione**: Le metriche, i valori e le specifiche sono stati estratti con successo dai file JSON, dimostrando la robustezza del sistema nell'analizzare contenuti complessi e arricchirli con informazioni contestuali.
- **Analisi approfondita**: La generazione di profili statistici ha fornito una comprensione dettagliata delle distribuzioni delle metriche, delle specifiche e dei valori, evidenziando tendenze e variazioni significative.
- **Standardizzazione dei dati**: L'allineamento terminologico ha garantito una rappresentazione uniforme delle informazioni, rendendo possibile un'analisi comparativa tra articoli scientifici di diversa origine e formato.

I risultati ottenuti dimostrano la capacità del sistema di analizzare dati tabulari in modo sistematico ed efficace, fornendo un framework solido per l'estrazione, la profilazione e l'allineamento delle informazioni. Questo rappresenta un passo importante verso l'automazione di processi complessi legati all'analisi di articoli scientifici, contribuendo a migliorare l'efficienza e l'affidabilità delle analisi condotte in ambito accademico e industriale.

6.1 Punti di forza e sfide incontrate

Tra i punti di forza principali del sistema sviluppato si evidenziano:

- **Modularità**: Il progetto è stato progettato in modo modulare, consentendo una facile estensione e integrazione di nuove funzionalità.
- **Flessibilità**: La capacità di gestire dati provenienti da diverse fonti e formati dimostra l'adattabilità del sistema a contesti eterogenei.
- **Precisione**: L'uso di modelli linguistici avanzati ha migliorato significativamente la precisione nell'estrazione e nell'allineamento delle informazioni.

Tuttavia, il progetto ha incontrato alcune sfide che rappresentano anche opportunità per sviluppi futuri:

- **Dati rumorosi**: Alcuni file JSON contenevano informazioni incomplete o incoerenti, che hanno richiesto passaggi di pre-elaborazione per garantire una corretta analisi.
- **Complessità semantica**: La variazione nella terminologia e nella struttura delle tabelle ha evidenziato la necessità di ulteriori miglioramenti nei modelli di allineamento semantico.
- **Limitazioni computazionali**: L'elaborazione di grandi dataset richiede risorse computazionali significative, che potrebbero essere ottimizzate con tecniche di parallelizzazione o con l'uso di infrastrutture cloud più avanzate.

6.2 Sviluppi Futuri

Nonostante i risultati positivi, ci sono diversi aspetti che potrebbero essere migliorati ed estesi. Alcuni possibili sviluppi futuri includono:

- **Miglioramento dell'accuratezza dell'estrazione**: L'integrazione di modelli linguistici di nuova generazione potrebbe migliorare la precisione nell'estrazione delle informazioni da contesti complessi.
- **Analisi semantica avanzata**: L'uso di modelli di embedding contestuali potrebbe migliorare l'allineamento terminologico, identificando sinonimi e concetti correlati con maggiore accuratezza.
- **Estensione a nuovi tipi di dati**: Il sistema potrebbe essere adattato per estrarre informazioni da formati diversi (ad esempio tabelle in formato Excel o documenti Word) o da contenuti non strutturati.
- **Visualizzazione dei risultati**: Sviluppare un'interfaccia grafica interattiva per visualizzare le distribuzioni delle metriche e delle specifiche, rendendo i risultati accessibili anche a utenti non tecnici.
- **Integrazione con altre risorse**: Collegare i risultati a database esistenti per arricchire ulteriormente l'analisi.

- **Automatizzazione completa**: Creare pipeline automatizzate per l'elaborazione end-to-end dei documenti, dalla classificazione iniziale all'output finale profilato e allineato.

In conclusione, il lavoro svolto rappresenta un importante passo verso l'automatizzazione dell'analisi di articoli scientifici, ma lascia ampio spazio per ulteriori miglioramenti e applicazioni in contesti più ampi.