Sistema di Data Discovery per file JSON

Alessio Marinucci & Riccardo Felici

`https://github.com/alemari7/Hw3_IDD`

Indice

1 Introduzione

In questo lavoro, l'obiettivo sarà quello di progettare e sviluppare un sistema di *data discovery* per la ricerca di risultati scientifici di interesse in un corpus di documenti. Il sistema permetterà agli utenti di eseguire query basate su termini che descrivono metriche e proprietà specifiche di risultati scientifici, restituendo un insieme ordinato di tabelle pertinenti. Per ottimizzare l'efficienza, le tabelle presenti negli articoli del corpus sono già state estratte e organizzate in un repository. Ogni tabella è accompagnata da metadati rilevanti, tra cui didascalie, paragrafi che citano la tabella e note a piè di pagina.
Il corpus di dati di riferimento per il progetto è accessibile all'indirizzo `all_tables`, che fornisce la base per il sistema da sviluppare.

Descrizione del Lavoro

Il sistema sarà in grado di:

- Interpretare query composte da termini che descrivono metriche e proprietà di risultati scientifici.
- Restituire un insieme ordinato di tabelle, sulla base di criteri di pertinenza e rilevanza.
- Analizzare e sfruttare i metadati associati a ogni tabella (didascalia, paragrafi di riferimento, note) per migliorare la qualità e la precisione delle risposte.

Le tabelle saranno classificate e ordinate utilizzando metodi che tengano conto della *Mean Reciprocal Rank (MRR)* e della *Normalized Discounted Cumulative Gain (NDCG)*, garantendo un'esperienza di ricerca efficace per l'utente.

Valutazione e Test

Il sistema sarà sottoposto a una fase rigorosa di valutazione utilizzando una metodologia che esamina la pertinenza dei risultati e la capacità di rispondere correttamente a query di varia complessità. Le metriche principali adottate per la valutazione saranno MRR e NDCG, che consentono di misurare la qualità delle risposte in termini di precisione e ordine.
I test saranno strutturati per analizzare i seguenti aspetti del sistema:

- **Copertura del Repository:** Saranno formulate query che verificano se tutte le informazioni estratte dal corpus (tabelle e relativi metadati) vengono adeguatamente interrogate e restituite.
- **Precisione e Rilevanza dei Risultati:** La valutazione si concentrerà sulla capacità del sistema di classificare correttamente le tabelle pertinenti in cima alla lista dei risultati. Questo sarà misurato utilizzando il *Mean Reciprocal Rank (MRR)*, che calcola la posizione del primo risultato rilevante rispetto alla query.
- **Richiamo e Pertinenza Relativa:** Per determinare l'efficacia del sistema nel considerare l'importanza relativa di più risultati, sarà utilizzata la metrica *Normalized*

Discounted Cumulative Gain (NDCG), che tiene conto sia della rilevanza dei risultati sia della loro posizione nella lista ordinata.

- **Prestazioni e Scalabilità:** Saranno monitorati i tempi di risposta del sistema per ogni query, con particolare attenzione alla gestione di grandi volumi di tabelle e query complesse.
- **Gestione di Query Complesse:** Verranno testate query che includono termini multipli e combinazioni complesse per verificare la capacità del sistema di gestire interazioni articolate e fornire risposte coerenti.
- **Robustezza e Gestione degli Errori:** Saranno simulate situazioni limite, come query vuote, termini inesistenti o metadati mancanti, per garantire che il sistema gestisca input problematici senza crash o errori significativi.

2 Pipeline del Progetto

Per lo svolgimento del progetto è stato necessario seguire una serie di passaggi, qui sotto descritti. Questa documentazione presenta una pipeline sistematica e ben organizzata per la realizzazione di un sistema di *Data Discovery* di file JSON, come illustrato in Figura 1.

2.1 Fasi della Pipeline

1. **Acquisizione dei Dati**
   (a) **Input dei File:** I file JSON vengono raccolti dalla directory designata (`all_tables`).
   (b) **Verifica dei File:** Controllare che i file siano accessibili, leggibili e correttamente formattati per evitare errori durante il parsing.
2. **Parsing dei Dati**
   (a) **Lettura dei File JSON:** Utilizzare la classe `FileReader` per leggere il contenuto dei file JSON.
   (b) **Parsing del Contenuto:** Convertire il contenuto dei file in oggetti JSON validi utilizzando librerie dedicate (ad esempio, `org.json.JSONObject`).
   (c) **Estrazione dei Dati Pertinenti:** Estrarre informazioni chiave come:
      - Contenuto delle tabelle.
      - Didascalie (*caption*) associate.
      - Note a piè di pagina (*footnotes*).
      - Riferimenti (*references*) o paragrafi che citano le tabelle.
3. **Indicizzazione**
   (a) **Preparazione dei Dati:** Pulire e trasformare i dati estratti in un formato adatto per l'indicizzazione.
   (b) **Utilizzo di un Analizzatore:** Applicare un analizzatore come il `StandardAnalyzer` di Apache Lucene per:
      - Tokenizzare il testo (suddividerlo in termini).
      - Rimuovere *stop words* e altri termini non significativi.
   (c) **Creazione dell'Indice:** Generare un indice strutturato, memorizzando campi distinti per ogni tabella (ad esempio, *table*, *caption*, *footnotes*, *references*) per un accesso rapido durante le ricerche.
4. **Querying**
   (a) **Input dell'Utente:** Ricevere query formulate dall'utente tramite console o interfaccia.
   (b) **Interpretazione della Query:** Analizzare la sintassi della query per identificare i campi su cui eseguire la ricerca (ad esempio, *caption*, *references* o *footnotes*).

(c) **Esecuzione della Query:** Interrogare l'indice utilizzando il motore di ricerca Lucene per ottenere risultati pertinenti.

5. **Restituzione dei Risultati**

   (a) **Formattazione dei Risultati:** Presentare i risultati della ricerca in un formato chiaro, includendo metadati come il nome del file sorgente, il contenuto della tabella, la didascalia, ecc.

   (b) **Visualizzazione:** Mostrare i risultati nella console o in un'interfaccia web progettata.

6. **Valutazione e Test**

   (a) **Testing delle Query:** Definire un insieme di query predefinite per valutare la capacità del sistema di restituire risultati pertinenti.

   (b) **Metriche di Valutazione:** Utilizzare metriche standard come:

   - *Mean Reciprocal Rank (MRR)* per valutare la posizione del primo risultato rilevante.
   - *Normalized Discounted Cumulative Gain (NDCG)* per misurare la qualità complessiva dei risultati ordinati.

   (c) **Analisi delle Prestazioni:** Monitorare i tempi di risposta per ogni query e valutare l'efficienza del sistema durante l'indicizzazione e la ricerca.

Figure 1: Pipeline dell'architettura utilizzata

3 Metriche e Valutazione

Per valutare le prestazioni del sistema, sono state adottate metriche standard che analizzano sia la rilevanza dei risultati sia la loro capacità di soddisfare le query dell'utente. Le metriche utilizzate sono descritte di seguito.

3.1 Mean Reciprocal Rank (MRR)

Nel sistema implementato, la MRR viene calcolata scorrendo i risultati restituiti dal motore di ricerca per identificare mediamente in che posizione si trova l'elemento più rilevante, ovvero quello con il valore di score più alto. All'interno del codice, dopo ogni query, vengono stampati gli indici dei documenti più rilevanti con tutte le informazioni richieste e in seguito viene richiesto all'utente di inserire gli indici dei documenti più rilevanti (anche in ordine sparso). Il sistema ricerca all'interno dell'array di indici l'elemento con score più alto e restituisce il valore di MRR come definito di seguito.

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

- Q rappresenta l'insieme delle query
- rank_i è la posizione (1-based index) del primo documento rilevante per la i-esima query.

Nel codice, la posizione del primo documento rilevante viene determinata tramite la funzione `calculateMRR`, che verifica se il documento con il punteggio più alto è presente tra quelli rilevanti. Un valore elevato di MRR indica che il sistema riesce a restituire rapidamente i risultati pertinenti. All'interno dell'esecuzione del programma, quando viene richiesto di inserire gli indici dei documenti più rilevanti, è necessario inserire sempre l'indice 0 dell'elemento con valore di score più alto, poichè è colui che definisce il valore di MRR calcolato ad ogni iterazione.

3.2 Normalized Discounted Cumulative Gain (NDCG)

La metrica NDCG valuta la qualità dell'intero ordinamento dei risultati rispetto alla loro rilevanza, assegnando un peso maggiore ai documenti rilevanti presenti nelle prime posizioni della lista. Nel sistema implementato, l'NDCG viene calcolata confrontando il *Discounted Cumulative Gain* (DCG) dei risultati restituiti con l'*Ideal Discounted Cumulative Gain* (IDCG) ottenuto ordinando idealmente i documenti in base alla loro rilevanza. La formula viene descritta in questo modo:

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$$

con:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}$$

$$\text{IDCG@}k = \sum_{i=1}^{k} \frac{\text{rel}_i^*}{\log_2(i+1)}$$

Di seguito definiamo le variabili:

- rel_i rappresenta la rilevanza del documento in posizione i nella lista restituita.
- rel_i^* rappresenta la rilevanza del documento in posizione i nell'ordinamento ideale.
- k è il numero massimo di risultati considerati.

Nel sistema, la funzione `calculateNDCG` calcola il DCG sommando i contributi dei documenti rilevanti sulla base della loro posizione e confronta il valore con l'IDCG, calcolato idealmente sui documenti rilevanti forniti dall'utente. Un valore di NDCG vicino a 1 indica che i risultati restituiti sono altamente rilevanti e ordinati in modo simile all'ordinamento ideale.

3.3 Esempi di query

In questa sezione vengono mostrati graficamente alcuni esempi di query svolte sul sistema. Da questi esempi di query, che sono state eseguite in ordine sulla console del programma, si può notare che il valore di MRR viene calcolato ad ogni iterazione per ogni query e viene utilizzato per calcolare il valore di MRR medio. Stesso discorso vale per l'altro tipo di metrica che è stata utilizzata, ovvero NDCG. Inoltre si può notare che aumentando la presione della query (quindi esplicitando una query combinata su cui fare l'analisi), il valore di score aumenta. In Tabella 1 vengono introdotti gli esempi di query che sono mostrati successivamente:

Query	**Dettagli**
Query 1	`caption: data`
Query 2	`references: model`
Query 3	`caption & references: learning, deep`

Table 1: Tabella con Query e Dettagli

Ricerca

1. **Query:** data
   Field: Caption
   Indici dei documenti rilevanti (separati da virgola): 4,3,2

   ## Metriche di valutazione:

   - **MRR (Mean Reciprocal Rank):** 0.3333333333333333
   - **NDCG (Normalized Discounted Cumulative Gain):** 0.6182885020492787
   - **Mean MRR:** 0.3333333333333333
   - **Mean NDCG:** 0.6182885020492787

   ## Risultati:

   DocId: 49006
   Score: 1.9990929

   Caption: *Table 5: Related research studies for Medical data properties in FL for medical applications, consisting of data partitions, data distribution (i.e., non-IID) characteristics, possible data privacy attacks, and data privacy protections.*

   Source File: `2405.13832.json`

2. **Query:** model
   Field: References
   Indici dei documenti rilevanti (separati da virgola): 1,2,3

   ## Metriche di valutazione:

   - **MRR (Mean Reciprocal Rank):** 0.5
   - **NDCG (Normalized Discounted Cumulative Gain):** 0.732826204777911
   - **Mean MRR:** 0.41666666666666663
   - **Mean NDCG:** 0.6755585612635349

   ## Risultati:

   DocId: 36695
   Score: 0.78918976

References: [The FLHub service consists of a web server and database with the communication of ";gRPC" (Figure ";2";). FLHub users are consist of model managers and learning participants with different roles like GitHub. In FLHub, the model manager is responsible for maintaining and merging models, and participants upload pull request like GitHub, the model manager examines the learning results contributed by each participant and issues a merge command if it is effective. "The following is a simple usage scenario of the FLHub service in four steps (Fig. ";2";)

Source File: 2202.06493.json

3. **Query:** learning,deep
Field: Caption & References
Indici dei documenti rilevanti (separati da virgola): 2,4,1

Metriche di valutazione:

- **MRR (Mean Reciprocal Rank):** 0.5
- **NDCG (Normalized Discounted Cumulative Gain):** 0.7122630665145961
- **Mean MRR:** 0.4444444444444444
- **Mean NDCG:** 0.6877933963472219

Risultati:

DocId: 45640
Score: 4.8331184

Caption: *Table 2: Comparison between deep meta-learning and vanilla meta-learning (deep version).*

References: ["DEML version vs. vanilla deep version. To validate that the improvements of DEML are not merely because of the deeper neural network and rescaled images, we also evaluate the deep versions of the previous approaches on MiniImagenet as mentioned in Section 4.2. We enlarge the meta-training dataset by merging together the original 64 classes of MiniImagenet and the 200 classes of ImageNet-200. The results are summarized in Table 2. It can be seen that simply enlarging the network and training dataset can not lead to a higher accuracy. DEML leverages the power of deep learning in a more principled way and achieves superior performance."]

Source File: 2202.06493.json

4 Conclusioni

Il progetto descritto in questo documento ha consentito di sviluppare un sistema di *data discovery* per la ricerca e la classificazione di tabelle scientifiche estratte da un corpus di documenti. Attraverso una pipeline ben definita e l'integrazione di strumenti per l'analisi e la valutazione, il sistema è in grado di rispondere in maniera efficace a query complesse, restituendo risultati pertinenti e ordinati sulla base di metriche standard come *Mean Reciprocal Rank (MRR)* e *Normalized Discounted Cumulative Gain (NDCG)*.

4.1 Risultati Principali

I principali risultati del progetto possono essere riassunti nei seguenti punti:

- **Rilevanza dei Risultati:** Il sistema ha dimostrato una buona capacità di identificare e restituire tabelle pertinenti, con valori medi di MRR e NDCG che confermano la qualità dell'ordinamento prodotto.
- **Gestione delle Query:** La pipeline implementata è in grado di processare query semplici e complesse, consentendo una ricerca accurata su campi specifici come *caption*, *references* e *footnotes* ed è anche in grado di processare query combinate su vari campi.
- **Robustezza:** I test condotti hanno evidenziato una buona gestione delle situazioni limite, come input incompleti o query non valide, garantendo la continuità operativa senza errori critici.

4.2 Sviluppi Futuri

In questa sezione vengono elencati alcuni possibili miglioramenti che potrebbero essere apportati al progetto di *data discovery*:

- **Integrazione con Tecniche di Apprendimento Automatico:** Implementare modelli di *neural networks* per migliorare la rilevanza dei risultati e automatizzare ulteriormente la classificazione dei documenti.
- **Valutazione Automatica:** Sviluppare un framework automatico per la valutazione delle metriche MRR e NDCG, riducendo la necessità di intervento manuale durante i test.
- **Inserimento di Strumenti di Embeddings:** Integrare modelli di embeddings, come *word embeddings* (ad esempio, Word2Vec o GloVe) o tecniche più recenti come BERT e Sentence Transformers, per rappresentare in modo semantico i contenuti delle tabelle e migliorare l'accuratezza nel confronto tra query e documenti.