Homework 2 - Ingegneria dei Dati
ElasticSearch: Sistema di indicizzazione e ricerca di file di testo

Alessandro Schmitt

Repository GitHub: **https://github.com/AlessandroSchmitt/elasticSearch**

1 Introduzione

Nel presente progetto, è implementato un sistema di indicizzazione e ricerca basato su Elasticsearch, progettato per analizzare e interrogare file di testo, contenenti trame di film. L'obiettivo è permettere la ricerca efficiente sia per nome del file (titolo del film) che per contenuto (trama), utilizzando analyzer appropriati e supportando query testuali inmesse dall'utente da console.

2 Ambiente di esecuzione

Hardware

- **Memoria RAM:** 7.6 GiB
- **Processore:** Intel(R) Core(TM) i5-4300M CPU @ 2.60GHz

Software

- **Sistema Operativo:** Ubuntu 5.15.0-139-generic
- **Docker:** Versione 28.1.1

Configurazioni Docker

Il progetto è stato eseguito utilizzando Docker per gestire i container.
La configurazione di Docker è stata effettuata con il seguente comando:

```
docker-compose up -d
```

Questo comando ha creato un ambiente di containerizzazione con il seguente servizio:

- **Elasticsearch**, in esecuzione su un container dedicato.

3 Dataset

Il dataset utilizzato per la realizzazione di questo homework è **Wikipedia Movie Plots**, scaricato da Kaggle (*https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots*). La prima fase di pre-processing ha previsto il subsampling del dataset, con lo scopo di mantenere circa 3000 righe su 35k, e la cancellazione delle colonne diverse da `Title` e `Plot`, in quanto non necessarie per questo caso d'uso. La colonna `Title` rappresenta i titoli dei film, mentre la colonna `Plot` rappresenta una descrizione di questi ultimi. Una volta preparato il dataset, ogni riga contenuta nel file `.csv` è stata convertita, attraverso apposito codice Python (prima parte del file `Indexer.ipynb`), in documenti `.txt`, dove il titolo è stato coerentemente utilizzato come nome del file e la descrizione del film come contenuto.

4 Indicizzazione dei File

Numero di File Indicizzati

- **File indicizzati:** 2.976 file

Processo di Indicizzazione

Pulizia dei Nomi dei File: per evitare problemi di compatibilità con il filesystem, è stata implementata una funzione Python che rimuove i caratteri non validi:

Listing 1: Funzione di sanitizzazione dei nomi dei file

```
def safe_filename(name):
    """Rimuove caratteri non validi da un nome di file."""
    name = re.sub(r'[\\/*?:"<>|]', "", name)
    name = re.sub(r"\s+", " ", name).strip()
    return name
```

Indicizzazione Bulk: l'indicizzazione è eseguita tramite l'API `helpers.bulk()` di Elasticsearch, con un timeout di 120 secondi. Ogni documento indicizzato segue la struttura seguente:

Listing 2: Struttura del documento indicizzato in Elasticsearch

```
{
  "_index": "moviesindex",
  "_source": {
    "title": "nome_film_senza_estensione",
    "content": "trama_del_film"
  }
}
```

Tempi di Indicizzazione

Il tempo di indicizzazione è stato misurato tramite un timer Python dalla libreria 'time':

Listing 3: Misurazione del tempo di indicizzazione

```
start = time.perf_counter()
success, _ = helpers.bulk(es.options(request_timeout=120), actions)
elapsed = time.perf_counter() - start
print(f"Tempo totale: {elapsed:.2f} secondi")
```

Con 2.976 documenti indicizzati:

- **Tempo totale:** 3.87 secondi.
- **Tempo medio stimato per documento:** circa 0.0013 secondi (1 millisecondo per documento);
- **Throughput medio:** circa 770 documenti al secondo.

  È importante notare che il tempo di indicizzazione di ElasticSearch dipende da diversi fattori. In particolare:

  - **Requests bulk e refresh:** L'uso di richieste di bulk ben dimensionate e un intervallo di refresh dell'indice adeguato consentono throughput più elevati.
  - **Allocazione hardware e I/O:** Le performance dipendono anche dalla macchina su cui Elasticsearch è eseguito. L'uso di SSD veloci, una buona allocazione di memoria e una CPU performante sono cruciali. La configurazione hardware può fare una grande differenza.

5 Analyzer

Per entrambi i campi (`title` e `content`) è stato scelto l'analyzer predefinito `english`, configurato nel mapping come segue:

Listing 4: Mapping dei campi `title` e `content`

```
{
  "mappings": {
    "properties": {
      "title": {
        "type": "text",
        "analyzer": "english",
        "search_analyzer": "english"
      },
      "content": {
        "type": "text",
        "analyzer": "english",
        "search_analyzer": "english"
      }
    }
  }
}
```

Motivazioni della Scelta dell'Analizzatore "english"

Per entrambi i campi `title` e `content` è stato scelto l'analizzatore predefinito "english" di Elasticsearch. Questa scelta è motivata da diverse ragioni legate alla natura dei dati da indicizzare e interrogare, ossia i titoli e le trame di film.

- **Rilevanza linguistica dei dati**: I file `.txt` da indicizzare contengono i titoli dei film nel campo `title` e le trame dei film nel campo `content`. Entrambi questi campi sono testi scritti in lingua inglese, una lingua che presenta caratteristiche morfologiche e sintattiche ben definite, come l'uso di stop words (parole comuni che non contribuiscono significativamente al significato del testo) e la possibilità di ridurre le parole alle loro radici (stemming). L'analizzatore "english" di Elasticsearch è progettato per gestire esattamente queste caratteristiche, rimuovendo automaticamente le stop words e riducendo le parole alle loro radici, migliorando la rilevanza e l'efficacia delle ricerche.
- **Funzionalità dell'analizzatore "english"**: L'analizzatore built-in "english" di Elasticsearch è composto da diversi componenti chiave:
  - **Tokenizer**: Il tokenizer utilizzato è il `standard tokenizer`, che suddivide il testo in termini (tokens) separando le parole usando spazi e punteggiatura. Questo è il tokenizer di default di Elasticsearch, e viene usato per la tokenizzazione base del testo.
  - **Filtri**: L'analizzatore "english" applica una serie di filtri specifici per la lingua inglese:
    - `lowercase filter`: converte tutti i termini in minuscolo, per evitare la distinzione tra "Movie" e "movie".
    - `stop filter`: rimuove le stop words, che sono parole molto comuni (come "the", "a", "is") che non aggiungono valore semantico alla ricerca e possono ridurre la qualità dei risultati.
    - `stemmer`: applica uno stemming specifico per l'inglese. Il filtro `english_stemmer` riduce le parole alla loro radice, ad esempio "running" diventa "run" o "better" diventa "good". Questo processo migliora la ricerca semantica, poiché "run" e "running" saranno trattati come la stessa parola.
  - **Ricerca (search analyzer)**: L'analizzatore usato durante la fase di ricerca (*search_analyzer*) è configurato anch'esso su "english", il che garantisce coerenza tra il processo di indicizzazione e quello di ricerca. In questo caso, il testo inserito nella query verrà trattato nello stesso modo, con tokenizzazione, conversione in minuscolo, rimozione delle stop words e stemming.
- **Uniformità della configurazione**: Poiché i campi `title` e `content` sono entrambi scritti in inglese e trattano argomenti simili (trame e titoli di film), è stato deciso di utilizzare lo stesso analizzatore per entrambi. Ciò garantisce coerenza nell'indicizzazione e nella ricerca, evitando eventuali discrepanze nei risultati tra i due campi. Se si utilizzassero analizzatori differenti, si potrebbero ottenere comportamenti inaspettati durante la ricerca, poiché un analizzatore potrebbe ridurre una parola a una forma diversa rispetto a un altro.

- **Semplicità e affidabilità dell'analizzatore built-in**: L'analizzatore "english" è uno degli analizzatori built-in di Elasticsearch, ben testato e ottimizzato per l'uso su testi in lingua inglese. Utilizzando un analizzatore predefinito, si beneficia di una configurazione stabile e di prestazioni ottimali senza la necessità di definire un analizzatore personalizzato. Questo approccio riduce la complessità e aumenta l'affidabilità dell'implementazione, permettendo di concentrarsi sul miglioramento di altri aspetti del sistema.

6 Sistema di Query

Stemming

Query: `adventure`
Se l'utente non specifica un campo, la ricerca verrà effettuata sia su title che su content.
Sono stati trovati 109 risultati per la ricerca della parola chiave "adventure". La ricerca ha considerato non solo la forma singolare "adventure", ma anche le sue varianti come "Adventure", "Adventurers" e "adventurers".... Questo comportamento riflette l'uso dell'analyzer "english" di Elasticsearch, che applica lo stemming per normalizzare le varianti grammaticali, trattando plurali e forme derivate come equivalenti durante la ricerca e che riduce a lower-case tutte le parole. Tale analyzer risulta quindi particolarmente adatta al contesto cinematografico senza la necessità di un analyzer personalizzato. Di seguito delle Query di Test per discutere ciascuno dei casi: plurali, stop-word, lower e upper-case...

Stop-Words

Query: `"the gang"`
Anche in questo caso la ricerca è effettuata sia su title che su content. Sono stati trovati 270 risultati. A differenza della query precedente, l'aggiunta dei double quote fa si che la ricerca sia sulla frase esatta. Il termine the è però una stop words che è automaticamente esclusa dal filtro cosi che vengano restituiti i documenti anche in assenza della parola "the". Lo stesso varrebbe per tutte le classiche parole in inglese considerate stopping words.

Upper-cases

Query: `title:YES`
La ricerca è effettuata solo su title. Sono stati trovati 2 risultati.
A differenza della query precedente, si discute la possibilità di recuperare documenti con un titolo che contenga la parola YES in tutte le sue forme: "yes", "Yes". Il comportamento atteso è coerente con quanto restituito dal sistema, dimostrando come il sistema risulti case-insensitive.

Plural

Query: `content:"wars"`
La ricerca è effettuata solo su content. Sono stati trovati 289 risultati.
Anche il tema dei plurali è sensibile a questo tipo di task, la ricerca di guerre deve restituire anche quei film che nella trama parlano distintamente di una gerra. Il comportamento è conforme a quello atteso. Dunque la forma plurale è correttamente ridotta alla forma base della parola.

Plural pt.2

Query: `content:men`
Il caso dei plurali potrebbe risultare critico quando ci sono forme irregolari.
Il comportamento è conforme anche in questo caso restituendo gli stessi documenti che sono restituiti con la query $content : man$.

Saxon genitive

Query: `title:"twilight's"`
viene correttamente restituito twilight's nonostante la presenza di *"'s"*, in quanto gestito dall'analyzer. In alcuni casi il genitivo sassone potrebbe necessitare di una gestione più forte: utilizzo del filtro *possessive*, non ritenuto necessario nella realizzazione di questo progetto.

Stemming x Semantics

Query: `title:warrior`
La ricerca è effettuata solo su title. Sono stati trovati 3 risultati.
C'è il rischio che "warrior" possa essere interpretato allo stesso modo che "war", ma un combattente in un dominio cinematografico potrebbe essere anche in un film per bambini o in altri contesti come per film che riprendono videogiochi o che parlano di sport ecc... Come ci si aspetta non si incorre in questa problematica, i film restituiti con tale ricerca non sono gli stessi restituiti con la ricerca se fatta nel seguente modo $title : war$.

Stemming x Semantics pt.2

Query: `title:tree`
La natura dell'analyzer presentata precedentemente può essere un beneficio come non. Di fatto, il caso di warrior e war è sottile, in alcuni contesti potremmo volere che la ricerca nelle due diverse maniere sia in Overlap almeno parziale, in altri no. Questa query per esempio restituisce un risultato non voluto, ovvero che la parola "tree" e la parola "three" e "trek" sono ridotte alla stessa radice, introducendo casi di false-positive. Non è un grosso problema se si considera la natura equilibrata tra efficacia ed efficienza riportata anche in casi real world.

accents

Query: `cafe`
i documenti restituiti sono tutti i documenti contenenti la parola cafe e café, evidenziano cosi la gestione intrinseca degli accenti che sono standarmente normalizzati. Evitando cosi che la presenza o meno di quest'ultimi incida sull'Output della ricerca. In alcuni casi potrebbe risultare opportuna l'aggiunta di un filtro "$asciifolding$".

7 Conclusioni e Considerazioni

Valutazione Complessiva del Sistema

Il sistema sviluppato ha dimostrato di essere **funzionale, scalabile e coerente** con gli obiettivi del progetto. L'integrazione tra la preparazione dei dati, la generazione dei file di testo e l'indicizzazione tramite Elasticsearch ha garantito un flusso operativo solido e ripetibile.

Prestazioni e Efficienza

Durante i test di indicizzazione e ricerca:

- il sistema ha gestito senza problemi **quasi 3.000 documenti**;
- le query restituivano risultati in tempi **inferiori al secondo**, anche per frasi complesse.

L'uso dell'analyzer `english` ha garantito un bilanciamento efficace tra **accuratezza linguistica** e **prestazioni**.

Punti di Forza

- **Semplicità di configurazione**: la pipeline di indicizzazione è composta da pochi passaggi essenziali e facilmente estendibili;
- **Riutilizzabilità**: il codice è modulare e può essere adattato ad altri dataset testuali;
- **Affidabilità**: l'uso delle API di Elasticsearch (`helpers.bulk`) assicura una gestione ottimizzata delle operazioni di scrittura.

Possibili Miglioramenti Futuri

Nonostante i risultati positivi, il sistema può essere ulteriormente migliorato attraverso:

- l'introduzione di analyzer multilingua o personalizzati per supportare dataset in lingue diverse dall'inglese;
- l'aggiunta di un'interfaccia grafica per la gestione e la visualizzazione dei risultati di ricerca;

Conclusione Finale

Il progetto ha permesso di comprendere in modo approfondito il funzionamento dei meccanismi di **indicizzazione e ricerca full-text** di Elasticsearch, evidenziando come un corretto preprocessing dei dati e la scelta di un analyzer appropriato siano fondamentali per ottenere risultati rilevanti e consistenti.