HW1 Ingegneria dei Dati – Indicizzazione con Elasticsearch

Link GitHub: `https://github.com/pietroyellow46/Data-Engineering`

Struttura del progetto ed esecuzione

Architettura generale del progetto

Il progetto ha come obiettivo l'indicizzazione e la ricerca di documenti testuali tramite **Elasticsearch**.

I principali file nel progetto sono:

- **indexer.py**: indicizza tutti i file `.txt` nella cartella `text/`, applicando l'analyzer italiano ai contenuti. Analizza inoltre i token dei campi e salva i risultati in un file JSON, includendo il tempo totale di indicizzazione.
- **searcher.py**: permette di effettuare ricerche interattive sull'indice, supportando query di tipo `term`, `match` e `match_phrase`.
- **mainApp.py**: applicazione principale che consente di scegliere interattivamente se indicizzare i file o eseguire ricerche, richiamando le funzioni di `indexer.py` e `searcher.py`.
- **config.py**: contiene tutte le configurazioni, come l'URL di Elasticsearch, il nome dell'indice, la cartella dei file testuali e le impostazioni per l'anteprima dei contenuti nelle ricerche.
- **indexerBenchmark.py**: script sperimentale utilizzato per confrontare le prestazioni dell'indicizzazione in modalità `bulk` e indicizzazione singola, misurando il tempo medio (mediana) su più run.

Metodi di esecuzione del progetto

Il motore di ricerca può essere facilmente eseguito rispettando i requisiti ed eseguendo i comandi riportati.

Requisiti:

- Python 3.10+
- Elasticsearch in esecuzione sul `http://localhost:9200`
- Le librerie Python `elasticsearch`, `datasets` e `pandas`

Comandi da eseguire nella cartella principale del progetto:

```
# Esecuzione
pyhton .\mainApp.py
```

In alternativa, per eseguire singolarmente i moduli:

```
# Indicizzazione e ricerca
pyhton .\indexer.py
pyhton .\indexerBenchmark.py
pyhton .\searcher.py
```

Analyzer scelti e motivazioni

Per l'indicizzazione dei campi sono stati scelti analyzer differenti in base al tipo di campo. Nel progetto sono stati utilizzati due approcci differenti di analisi testuale per i campi indicizzati, con l'obiettivo di sperimentare le principali tipologie di Analyzer e comprenderne l'impatto sulle query e sui risultati.

1. Campo nome

Per il campo `nome` è stato utilizzato l'analyzer **keyword**. Questo analyzer non effettua tokenizzazione né conversione in minuscolo: il valore del campo viene trattato come un singolo token. La scelta è motivata dal fatto che i nomi dei file devono essere confrontati *esattamente* come inseriti, distinguendo tra maiuscole, minuscole e punteggiatura. In questo modo, la ricerca `nome:index.txt` restituirà risultati solo in caso di corrispondenza esatta. Tale scelta permette di distinguere con precisione documenti diversi anche in presenza di maiuscole/minuscole, simulando un comportamento realistico di ricerca di file in un filesystem.

2. Campo contenuto

Per il campo `contenuto` è stato scelto l'analyzer **standard** (nella versione `italian`). Questo analyzer esegue la tokenizzazione in parole, converte in minuscolo e rimuove la punteggiatura. È quindi adatto per le ricerche testuali, permettendo il matching anche indipendentemente dalle maiuscole o dalle variazioni morfologiche delle parole.

- tokenizzazione in base a spazi e punteggiatura
- rimozione delle stopword in lingua italiana
- conversione di tutto il testo in minuscolo

Test di indicizzazione

Per valutare le prestazioni dell'indicizzazione sono stati effettuati test sperimentali utilizzando il programma di benchmark sviluppato (`indexerBenchmark.py`), che confronta due modalità di inserimento dei documenti in Elasticsearch:

- **Indicizzazione Bulk**, in cui più documenti vengono inseriti in un'unica operazione.
- **Indicizzazione Singola**, in cui ogni documento viene inviato separatamente.

Il dataset utilizzato per i test è **"Abirate/french_book_reviews"**, disponibile su **Hugging Face**. Da tale dataset sono state selezionate **4.600 righe**, ognuna contenente un titolo del libro e una recensione del lettore, indicizzate rispettivamente nei campi `nome` e `contenuto`.

Ogni esperimento è stato ripetuto **10 volte** per ottenere una stima più stabile dei tempi medi, calcolando la **mediana** delle durate.

Risultati dell'indicizzazione Bulk

```
Starting experiment: Bulk Indexing
Run 1/10 - Documents indexed: 4600, Time: 4.727s
Run 2/10 - Documents indexed: 4600, Time: 2.140s
Run 3/10 - Documents indexed: 4600, Time: 1.795s
Run 4/10 - Documents indexed: 4600, Time: 1.363s
Run 5/10 - Documents indexed: 4600, Time: 3.818s
Run 6/10 - Documents indexed: 4600, Time: 2.125s
Run 7/10 - Documents indexed: 4600, Time: 1.756s
Run 8/10 - Documents indexed: 4600, Time: 1.448s
Run 9/10 - Documents indexed: 4600, Time: 1.545s
Run 10/10 - Documents indexed: 4600, Time: 1.502s

Results Bulk Indexing
Times: [4.727, 2.14, 1.795, 1.363, 3.818, 2.125, 1.756, 1.448, 1.545, 1.502]
Median time: 1.776 seconds
```

Risultati dell'indicizzazione Singola

```
Starting experiment: Single Indexing
Run 1/10 - Documents indexed: 4600, Time: 271.154s
Run 2/10 - Documents indexed: 4600, Time: 291.091s
Run 3/10 - Documents indexed: 4600, Time: 287.035s
Run 4/10 - Documents indexed: 4600, Time: 297.510s
```

```
Run 5/10 - Documents indexed: 4600, Time: 294.765s
Run 6/10 - Documents indexed: 4600, Time: 295.575s
Run 7/10 - Documents indexed: 4600, Time: 298.528s
Run 8/10 - Documents indexed: 4600, Time: 295.807s
Run 9/10 - Documents indexed: 4600, Time: 294.640s
Run 10/10 - Documents indexed: 4600, Time: 287.147s

Results Single Indexing
Times: [271.1, 291.0, 287.0, 297.5, 294.7, 295.5, 298.5, 295.8, 294.6, 287.1]
Median time: 294.70 seconds
```

Confronto finale

```
Final Comparison
Bulk Median: 1.776 s
Single Median: 294.703 s
```

Dai risultati emerge una **netta differenza di prestazioni** tra le due modalità: l'indicizzazione **Bulk** risulta circa **165 volte più veloce** rispetto all'indicizzazione singola. Questo conferma l'efficacia dell'uso delle API Bulk di Elasticsearch per carichi di lavoro di grandi dimensioni.

Query di test e risultati

Le seguenti query sono state utilizzate per verificare il corretto funzionamento del motore di ricerca, esplorando tutti i casi d'uso principali: ricerche esatte, errori di maiuscole, frasi tra virgolette e query malformate. Di seguito vengono riportate le query eseguite e i rispettivi risultati.

Query 1 – Ricerca per nome

Descrizione: la ricerca sul campo `nome` utilizza una query che non viene analizzata. Trova corrispondenza solo se il valore coincide esattamente con quello indicizzato.

Query: `nome:index.txt`

Documenti trovati:

1. **index.txt** (score=1.99) - preview: L'indicizzazione è il processo cruciale nei sistem...

Query 2 – Ricerca per nome con maiuscole errate

Descrizione: Il campo `nome` non viene elaborato dallo `KeywordAnalyzer`, quindi la ricerca è case-sensitive. "indEX.txt" non coincide con "index.txt" e non produce risultati.

Query: `nome:indEX.txt`

Risultato: Nessun documento trovato.

Query 3 – Ricerca per nome con errore

Descrizione: Il nome del file reale è "lucene.txt". Poiché la ricerca richiede una corrispondenza esatta, l'errore di battitura non fa produrre al motore di ricerca alcun risultato.

Query: `nome:lucenee.txt`

Risultato: Nessun documento trovato.

Query 4 – Ricerca testuale semplice su contenuto

Descrizione: Una semplice ricerca testuale sul contenuto. Tutti i documenti che contengono la parola "crowdsourcing" vengono trovati

Query: `contenuto:crowdsourcing`

Documenti trovati:

1. **crowdsourcing.txt** (score=2.02) - preview: Il Crowdsourcing è un modello che sfrutta la capac...

Query 5 – PhraseQuery semplice su contenuto

Descrizione: Viene effettuata una semplice `PhraseQuery`, che cerca corrispondenze esatte tra la query effettuata e ciò che c'è nell'indice.

Query: `contenuto:"crowdsourcing"`

Documenti trovati:

1. **crowdsourcing.txt** (score=2.02) - preview: Il Crowdsourcing è un modello che sfrutta la capac...

Query 6 – Ricerca testuale con più termini

Descrizione: La query implicitamente mette in OR tutti i termini che formano la query. Tutti i documenti che contengono i termini "engineer" o "crowdsourcing" vengono restituiti.

Query: `contenuto:engineer crowdsourcing`

Documenti trovati:

1. **crowdsourcing.txt** (score=2.02) - preview: Il Crowdsourcing è un modello che sfrutta la capac...
2. **data_engineering.txt** (score=2.02) - preview: Il Data Engineering è la disciplina che si occupa ...

Query 7 – PhraseQuery con variazioni di maiuscole

Descrizione: La query viene pre-processata con lo stesso `Analyzer` usato per il campo su cui si effettua la query e, di conseguenza, viene, tra le altre cose, normalizzata, portando tutto in minuscolo.

Query: `contenuto:"CROWDSOURCING"`

Documenti trovati:

1. **crowdsourcing.txt** (score=2.02) - preview: Il Crowdsourcing è un modello che sfrutta la capac...

Query 8 – PhraseQuery con più parole

Descrizione: Una `PhraseQuery` con più parole si comporta allo stesso modo di una con un singolo termine: viene semplicemente cercata la corrispondenza esatta con una stringa come nel caso del singolo termine che, in questo caso, è però formata da più termini.

Query: `contenuto:"pipeline di dati"`

Documenti trovati:

1. **data_engineering.txt** (score=2.02) - preview: Il Data Engineering è la disciplina che si occupa ...

Query 9 – PhraseQuery con errore

Descrizione: Manca la preposizione “di”; L’assenza della sequenza esatta impedisce la corrispondenza, e la query non restituisce risultati.

Query: `contenuto:"pipeline dati"`

Risultato: Nessun documento trovato.

Query 10 – Campo inesistente

Descrizione: Il campo “campoFinto1” non esiste nell’indice. Elastisearch gestisce il caso mandando un errore di sintassi poichè il primo campo che trova non è tra i campi su cui è costruito l’indice.

Query: `campoFinto1:campoFinto2`

Risultato: Invalid syntax. Use nome:document1 or contenuto:word or contenuto:”phrase”

Query 11 – Sintassi errata

Descrizione: L’input non rispetta il formato previsto (manca “:” come separatore). Il software riconosce la sintassi errata e mostra un messaggio di errore.

Query: `contenuto-ciao`

Risultato: Invalid syntax. Use nome:document1 or contenuto:word or contenuto:”phrase”