Relazione Homework 2: Indicizzazione Documenti TXT con Elasticsearch

di Mario Alessandro, n°Matricola: 547297

3 Novembre 2025

1 Introduzione

Il presente documento descrive la soluzione implementata per l'Homework 2 del corso di Ingegneria dei Dati, focalizzato sull'indicizzazione di file di testo (`.txt`) e sull'implementazione di un sistema di interrogazione basato su **Python** e **Elasticsearch**. Il sistema indicizza due campi distinti: il nome del file e il contenuto del file.

1.1 URL del Progetto

Il codice sorgente è disponibile al seguente indirizzo:
`https://github.com/ArthexTheKing/Ingegneria-dei-Dati/tree/master/Homework%202/di%20Mario%20Alessandro`

2 Configurazione e Analisi

2.1 Analizzatori Scelti

Sono stati definiti due analizzatori, personalizzati e standard, per i campi `nome` e `contenuto`, rispettando il vincolo di non utilizzare il tipo `keyword`.

Tabella 1: Analizzatori Utilizzati e Motivazioni

Campo	Analizzatore	Componenti	Motivazione
`nome`	`filename_analyzer` (Personalizzato)	Tokenizer `whitespace`, Filtri `lowercase` e `italian_stop`	Il tokenizer `whitespace` è stato scelto per la sua semplicità. I nomi dei file sono stati modificati per utilizzare lo **spazio** come delimitatore principale, garantendo il corretto funzionamento del tokenizer e delle query.
`contenuto`	`italian` (Standard)	Tokenizer `standard`, Filtri `lowercase`, `italian_stop` e `italian_stemmer`	L'analizzatore **italian** è la scelta più appropriata per il contenuto in lingua italiana. L'uso dello **stemming** (incluso) aumenta la tolleranza e il *recall* della ricerca.

2.2 Dati di Indicizzazione e Prestazioni

Sono stati indicizzati 10 file di testo, con contenuto corposo e tematica sul Deep Learning. Il tempo di indicizzazione è stato misurato utilizzando la libreria `time` di Python.

- **Numero di file indicizzati:** 10
- **Tempo totale di indicizzazione: 0.5321** secondi

3 Interrogazione del Sistema

Il programma di interrogazione legge le query dalla console e utilizza la sintassi della `query_string_query` di Elasticsearch, rispettando il formato richiesto (`campo:termine` o `campo:"phrase"`).

3.1 Query di Test (10 Esempi)

Il sistema è stato testato con le seguenti 10 query, ciascuna progettata per verificare specifici aspetti del sistema di ricerca e degli analizzatori.

Tabella 2: Query di Test e Obiettivi

#	Query	Tipo di Ricerca	Obiettivo del Test
1	`nome:backpropagation`	Termine Singolo (Nome)	Verifica la ricerca su un singolo termine nel campo `nome`.
2	`nome:"reti neurali"`	Phrase Query (Nome)	Testare la ricerca di una frase esatta nel nome.
3	`contenuto:classificazione`	Termine Singolo (Contenuto)	Testare l'efficacia dello **stemming** nell'analizzatore `italian`.
4	`contenuto:"dati sequenziali"`	Phrase Query (Contenuto)	Verificare la ricerca di una sequenza esatta di parole nel contenuto.
5	`contenuto:imaging OR contenuto:raggi`	Booleana OR (Contenuto)	Verificare la capacità di recuperare documenti che contengono almeno uno dei termini.
6	`contenuto:sfide AND contenuto:black`	Booleana AND (Contenuto)	Verificare l'AND implicito tra concetti nello stesso documento (`sfide` e `black box`).
7	`nome:tensorflow OR nome:pytorch`	Booleana OR (Nome)	Testare la ricerca disgiuntiva sui titoli dei file.
8	`nome:"reti generative" AND contenuto:generatore`	Ricerca Mista (Nome e Contenuto)	Combinare la ricerca su entrambi gli indici.
9	`contenuto:sequenze OR nome:trasformatore`	Ricerca Mista OR	Ricerca alternativa tra un concetto nel contenuto e una parola nel titolo.
10	`nome:apprendimento*`	Wildcard (Nome)	Testare la ricerca con caratteri jolly (`*`) sul nome del file.