

# Ingegneria dei Dati 2025/2026

## Homework 2

### Sistema di Information Retrieval basato su Apache Lucene

Studente: Alessandro Cavina

## URL del Progetto

Repository GitHub: <https://github.com/Alessandro2801/IR-System-with-Apache-Lucene>

## Introduzione

L'obiettivo del progetto è la realizzazione di un sistema di **Information Retrieval** sviluppato in linguaggio **Java** mediante l'utilizzo della libreria **Apache Lucene**. Il sistema è stato concepito per indicizzare una collezione di documenti testuali, con l'obiettivo di consentire una ricerca efficiente, mirata e scalabile all'interno di un corpus di dati non strutturati.

In particolare, il progetto si propone di costruire un motore di ricerca capace di analizzare e organizzare una serie di file `.txt` presenti in una directory locale, associando a ciascun documento due principali campi informativi: il *titolo* e il *contenuto*. L'indicizzazione di tali campi consente di rendere possibile l'esecuzione di query personalizzate, le quali vengono interpretate tramite un'interfaccia utente da console e utilizzate per restituire, in ordine di rilevanza, i documenti più pertinenti rispetto ai termini di ricerca forniti.

In sintesi, il progetto consente di trasformare una semplice collezione di file testuali non strutturati in un sistema di ricerca indicizzato, efficiente e interattivo, dimostrando le potenzialità di Apache Lucene nella gestione, indicizzazione e recupero di informazioni testuali su larga scala.

## Generazione del Dataset

Il dataset utilizzato per il progetto è stato scaricato dal portale **Kaggle** e contiene una collezione di film con le relative informazioni testuali. Il file originale, in formato `.csv`, comprende diverse colonne descrittive tra cui il **Title** (titolo del film) e il **Plot** (trama), che rappresentano i due campi di principale interesse per la costruzione del sistema di *Information Retrieval*.

Il dataset presenta le seguenti caratteristiche principali:

- Numero totale di record: *35 000 film*;
- Numero di colonne: *8 : {Year, Title, Origin, Director, Cast, Genre, Wiki Page, Plot}*;
- Campi di interesse: **Title** (titolo) e **Plot** (trama);
- Lingua originale: inglese.

Prima di procedere con la fase di indicizzazione, il dataset è stato preprocessato mediante uno script **Python**, con l'obiettivo di:

- rilevare ed eliminare eventuali duplicati basandosi sul campo **Title**, per garantire l'unicità dei file generati;
- tradurre in lingua italiana i campi **Title** e **Plot** utilizzando la libreria `deep_translator` (servizio *GoogleTranslator*);
- creare, per ogni record univoco, un file **.txt** contenente la trama del film tradotta.

Dunque per ciascun record univoco del dataset vengono prelevati i campi **Title** e **Plot**, che rappresentano rispettivamente il titolo e la trama del film. Entrambi i campi vengono tradotti in lingua italiana e successivamente utilizzati per la creazione di un documento testuale. In particolare, il titolo tradotto viene impiegato come nome del file, mentre la trama costituisce il contenuto del file stesso. Ogni documento così generato viene salvato all'interno della directory `files/`, che rappresenta il corpus di documenti da indicizzare nella fase successiva.

Nel caso in cui si verifichi un errore durante la traduzione di un record, il file corrispondente non viene creato, e lo script procede automaticamente con l'elaborazione del record successivo. In questo modo si garantisce la consistenza e l'affidabilità del dataset finale destinato alla fase di indicizzazione.

Il risultato finale di questa fase è una collezione di **documenti testuali univoci e in lingua italiana**, pronti per essere indicizzati tramite la libreria **Apache Lucene**.

## Implementazione del Sistema di Indicizzazione

La fase di indicizzazione è stata realizzata in linguaggio **Java** utilizzando la libreria **Apache Lucene**, che costituisce uno degli strumenti più diffusi e potenti per la costruzione di motori di ricerca testuali. L'obiettivo di questa fase è trasformare la collezione di file **.txt** presenti nella directory `files/` in un indice ricercabile, in cui ciascun documento è rappresentato da due campi principali: **title** e **content**.

Per ogni file individuato nella directory, il sistema crea un oggetto `Document` di Lucene, al quale vengono aggiunti:

- il campo **title**, che contiene il nome del file (ovvero il titolo del film);
- il campo **content**, che contiene il testo della trama.

Ciascun campo è stato gestito mediante un **Analyzer** specifico, progettato per ottimizzare il processo di tokenizzazione e indicizzazione in funzione delle caratteristiche del testo.

## Analyzer per il campo title

Per il campo `title` è stato definito un **analyzer personalizzato**, costruito tramite la classe `CustomAnalyzer` di Lucene, come segue:

```
Analyzer titleAnalyzer = CustomAnalyzer.builder()
    .withTokenizer(WhitespaceTokenizerFactory.class)
    .addTokenFilter(WordDelimiterGraphFilterFactory.class,
        "generateWordParts", "1",
        "catenateNumbers", "1",
        "catenateWords", "1",
        "splitOnCaseChange", "1")
    .addTokenFilter(LowerCaseFilterFactory.class)
    .addTokenFilter(ASCIIFoldingFilterFactory.class)
    .build();
```

Questo analyzer è stato progettato per gestire in modo accurato le peculiarità tipiche dei titoli cinematografici, che possono contenere numeri, trattini o variazioni di maiuscole e minuscole. In particolare:

- la `WordDelimiterGraphFilterFactory` consente di separare e concatenare parole e numeri (es. `Spider-Man` → `spider`, `man` e `spiderman`);
- il filtro `LowerCaseFilterFactory` converte tutti i token in minuscolo, rendendo la ricerca *case-insensitive*;
- il filtro `ASCIIFoldingFilterFactory` elimina gli accenti e normalizza i caratteri speciali (es. `Maestà` → `maesta`).

A differenza di altri analyzer predefiniti, in questo caso non è stato applicato alcun filtro di stopword. Questa scelta è motivata dal fatto che il titolo di un film rappresenta un'informazione fortemente identificativa e semantica: anche parole comuni o grammaticalmente poco significative possono contribuire a distinguere un titolo da un altro. Pertanto, la rimozione delle stopword avrebbe potuto compromettere la precisione dei risultati di ricerca.

## Analyzer per il campo content

Il campo `content`, che rappresenta la trama del film, è stato invece indicizzato utilizzando l'analyzer `ItalianAnalyzer` fornito da Lucene. Questo analyzer combina un insieme di componenti ottimizzati per la lingua italiana, tra cui:

- tokenizzazione standard per la separazione delle parole;
- rimozione delle **stopword** più comuni della lingua italiana (es. “il”, “la”, “di”, “e”);
- applicazione di uno **stemmer** leggero basato sull'algoritmo Snowball, per ricondurre le parole alle loro forme radice mantenendo la correttezza morfologica.

Questa configurazione consente di ottenere una rappresentazione semantica più compatta e coerente del contenuto testuale, migliorando la qualità delle corrispondenze durante le ricerche.

Nel complesso, la combinazione dei due analyzer, uno specifico per i titoli e uno linguistico per i contenuti, garantisce un equilibrio tra precisione e flessibilità, permettendo di

valorizzare sia gli aspetti identificativi del titolo sia la ricchezza semantica delle trame. L'uso combinato dei due analyzer consente di mantenere la precisione nella ricerca per titolo, pur garantendo una maggiore flessibilità semantica nelle ricerche full-text sulla trama.

## Interfaccia Utente su Console

È stata implementata una semplice interfaccia a linea di comando che consente all'utente di eseguire query personalizzate sull'indice. Il sistema interpreta le query dell'utente, permettendo di specificare il campo da ricercare (es. **titolo** o **trama**) o di eseguire ricerche generiche su entrambi i campi.

Le query sono gestite tramite un **QueryParser** di Lucene, che consente di combinare più termini all'interno della stessa ricerca. In particolare, il sistema interpreta i termini separati da spazi come connessi da un **OR logico**, restituendo tutti i documenti che contengono almeno uno dei termini specificati. Inoltre, quando i termini vengono racchiusi tra virgolette doppie (" "), il sistema costruisce una **PhraseQuery**, ricercando la sequenza esatta di parole nella forma specificata.

Esempi di input:

```
Query > titolo inception
Query > trama "guerra mondiale"
```

Per ciascuna query, il sistema utilizza un **QueryParser** associato all'analyser del campo corrispondente e restituisce una lista ordinata (ranked list) di documenti rilevanti, mediante la tecnica **tf-idf**, mostrando:

- il titolo del film recuperato;
- un frammento della trama;
- lo score associato dal ranking di Lucene.

```
Trovati 1 documenti (ricerca in 5 ms):
20.000 leghe sotto i mari (docId=174, score=11,2664)
    Una strana gigantesca "creatura marina" imperversa nei mari. La
        nave da guerra americana Abraham Lincoln viene inviata a
            indagare, ma viene speronata dalla "creatura" che si scopre
                essere il Nautilus,...
```

Questa modalità di interrogazione offre un'interazione flessibile e naturale con il sistema, permettendo di esprimere sia ricerche più ampie (basate su corrispondenze parziali dei termini) sia ricerche più precise mediante sequenze testuali esatte.

## Statistiche sull'Indicizzazione

Per valutare le prestazioni del sistema di indicizzazione, sono state raccolte alcune statistiche relative alla fase di creazione dell'indice tramite Apache Lucene. Durante questa fase, per ogni file .txt presente nella directory specificata, è stato creato un documento Lucene contenente i campi **title** e **content**.

Il tempo complessivo di indicizzazione è stato stimato misurando l'intervallo tra l'avvio e il termine del processo di creazione dell'indice, utilizzando le funzioni di temporizzazione del linguaggio Java. In particolare, la misura è stata effettuata con le seguenti istruzioni:

```

long start = System.currentTimeMillis();
Indexer.createIndex(docsPath, indexPath);
long end = System.currentTimeMillis();

```

La differenza tra i due valori (`end - start`) fornisce il tempo totale di esecuzione in millisecondi, successivamente convertito in secondi per una più agevole interpretazione.

Parametro	Valore
Numero di documenti indicizzati	<b>3000</b>
Directory dei documenti	<b>files/</b>
Tempo di indicizzazione	<b>1.47 secondi</b>

Tabella 1: Statistiche relative al processo di indicizzazione.

Come mostrato in tabella, l'indicizzazione di 3000 documenti è stata completata in circa 2.3 secondi, evidenziando l'efficienza del motore di indicizzazione di Lucene. Questo risultato conferma la capacità della libreria di gestire corpus di migliaia di documenti con tempi di elaborazione estremamente contenuti.

## Test e query di verifica del sistema

Per testare il corretto funzionamento del sistema, sono state formulate 10 query rappresentative, progettate per valutare la qualità dell'indicizzazione e della ricerca nei campi titolo e trama.

### 1. Query : titolo "alice nel paese delle meraviglie"

Questa query utilizza una *PhraseQuery*, ossia una ricerca di una sequenza esatta di termini. L'obiettivo è verificare che Lucene restituisca solo i documenti in cui la frase completa appare nel titolo, mostrando il corretto funzionamento del parsing e della tokenizzazione delle frasi tra virgolette.

```

Query > titolo "alice nel paese delle meraviglie"
Query interpretata: title:"alice nel paese delle meraviglie"
● Trovati 1 documenti (ricerca in 61 ms):
● Alice nel paese delle meraviglie (docId=2725, score=11,6433)
    Alice segue un grande coniglio bianco nella "tana del coniglio". Trova una piccola porta. Quando trova una bottiglia con l'etichetta "Bevimi", lo fa
    si rimpicciolisce, ma non abbastanza per passare ...

```

### 2. Query : titolo conte montecristo

Questa query testa il comportamento predefinito del *QueryParser*, che effettua un **OR logico** tra i termini non virgolettati. È utile per verificare che vengano recuperati tutti i documenti che contengono almeno uno dei termini indicati, confermando la corretta configurazione del parser.

```

Query > titolo conte montecristo
Query interpretata: title:conte title:montecristo
● Trovati 3 documenti (ricerca in 8 ms):
● Il Conte di Montecristo (docId=1366, score=5,9726)
    Nel 1815 una nave mercantile francese fa tappa all'Isola d'Elba. Una lettera di Napoleone in esilio viene consegnata al capitano della nave perché la
    consegni a un uomo a Marsiglia. Prima di morire di...
● Il conte di Chicago (docId=1371, score=2,9863)
    Per rimediare alle cattive azioni del suo passato, Robert "Silky" Kilmount, ex contrabbandiere di Chicago che ha aperto la propria distilleria legale
    assume Quentin "Doc" Ramsey come manager della su...
● Il figlio di Montecristo (docId=2957, score=2,9863)
    Nel 1865 il generale proletario Gurko Lanen (George Sanders) diventa il dittatore dietro le quinte del Granducato di Lichtenburg situato nei Balcani.
    Gurko sopprime il clero e la libertà di stampa e...

```

### 3. Query : titolo "la"

Questa query è stata eseguita per mostrare che il sistema non effettua *stopword removal* sul campo **title**. In questo modo vengono restituiti tutti i film che iniziano con “La”, comportamento desiderato in quanto il titolo è un campo altamente informativo e non deve essere normalizzato come un testo generico.

```
Query > titolo "La"
Query interpretata: title:la
🔍 Trovati 401 documenti (ricerca in 3 ms):
● La lotta per la vita (docId=232, score=1,1043)
    Al City Hospital una giovane stagista muore di giovane madre in sala parto. Molto preoccupato di aver trascurato un fatto che avrebbe potuto prevenire la morte, iniziò a frequentare una clinica di mat...
● La lotta per la libertà (docId=1294, score=1,1043)
    Il film si apre in una città al confine con il Messico. Nel saloon locale è in corso una partita di poker. Uno dei giocatori bara e viene ucciso da un altro giocatore, un messicano di nome Pedro. Nel ...
● La Lettera (docId=469, score=1,0949)
    Annoia e solitaria nella piantagione di gomma del marito, Leslie Crosbie ha un amante, Geoffrey Hammond. Alla fine, però, si stanca di lei e prende un'amante cinese, Li-Ti. Quando Leslie lo scopre, ...
● La chiave (docId=477, score=1,0949)
    Il capitano Bill Tennant (William Powell) è un ufficiale britannico di stanza a Dublino nel 1920. Tennant ha una storia con Norah, la moglie del suo socio, l'ufficiale dell'intelligence britannica, il...
● La Mongolfiera (docId=543, score=1,0949)
    Un giovane (Keaton) ha una serie di incontri in un'area di divertimenti, molto simile a Coney Island, finché non si imbatte in un gruppo di uomini che preparano una mongolfiera per il lancio. Il giova...
```

### 4. Query : titolo 20000

Con questa query si verifica la corretta gestione dei numeri e della punteggiatura, dimostrando che il sistema riconosce equivalenti forme come “20000” e “20.000”. Tale funzionalità è garantita dal filtro **WordDelimiterGraphFilterFactory** configurato nell’analyzer personalizzato per il campo titolo.

```
Query > titolo 20000
Query interpretata: title:20000
🔍 Trovati 1 documenti (ricerca in 0 ms):
● 20.000 leghe sotto i mari (docId=388, score=2,6108)
    Una strana gigantesca "creatura marina" imperversa nei mari. La nave da guerra americana Abraham Lincoln viene inviata a indagare, ma viene speronata dalla "creatura" che si scopre essere il Nautilus,...
```

### 5. Query : titolo spider man

Questa query serve a mostrare che il sistema riconosce correttamente le varianti morfologiche del titolo, come “Spider-Man” o “spiderman”. Il risultato positivo conferma la corretta applicazione dei filtri di lowercasing, tokenizzazione per delimitatori e rimozione dei caratteri speciali.

```
Query > titolo spider man
Query interpretata: title:spider title:man
🔍 Trovati 3 documenti (ricerca in 1 ms):
● Spider-Man (docId=2440, score=7,8138)
    Peter Parker, liceale, vive con la zia May e lo zio Ben ed è un emarginato della scuola. Durante una gita scolastica, visita un laboratorio di genetica con il suo amico Harry Osborn e l'interesse amor...
● F-Man (docId=284, score=3,6763)
    Johnny Dime va in California determinato a diventare un agente governativo. Invece finisce per diventare un idiota della soda, poi mente alla fidanzata Molly Carter quando lei lo segue a ovest, sostiene...
● Iron Man (docId=288, score=3,6763)
    Dopo che il pugile leggero Kid Mason (Ayers) perde il suo incontro di apertura, la moglie Rose (Harlow), cercatrice d'oro, lo lascia per Hollywood. Nata di lei in giro, Mason si allena seriamente e...
```

### 6. Query : titolo maesta

Questa query verifica la gestione degli accenti, mostrando che la parola “maestà” viene riconosciuta anche se digitata senza accento. Il comportamento corretto è reso possibile dal filtro **ASCIIFoldingFilterFactory**, che consente il matching tra caratteri accentati e non.

```
Query > titolo maesta
Query interpretata: title:maesta
🔍 Trovati 2 documenti (ricerca in 0 ms):
● Sua Maesta, l'americano (docId=2074, score=2,9863)
    Come descritto in una rivista di cinema, Bill (Fairbanks), le cui buffonate da far rizzare i capelli lo hanno reso oggetto di conversazione a New York, decide di lasciare la metropoli dopo che un...
● Sua Maesta, lo Spaventapasseri di Oz (docId=2940, score=2,4353)
    King Krewl (Raymond Russell) è un crudele dittatore della Città di Smeraldo nella Terra di Oz. Desidera sposare sua figlia, la principessa Gloria (Gloria Reed), con un vecchio cortigiano di nome Googl...
```

## 7. Query : titolo montecristo9

L'obiettivo di questa query è testare la robustezza del sistema in presenza di piccole imprecisioni di input (errori ortografici o di battitura). Sebbene Lucene non implementi nativamente la fuzzy search nel parser di base, l'analisi mostra che l'individuazione tokenizzata e lo stemming possono in alcuni casi compensare parzialmente errori minori.

```
Query > titolo montecristo9
Query interpretata: title:montecristo title:9
● Trovati 2 documenti (ricerca in 0 ms):
● Il Conte di Montecristo (docId=1366, score=2,9863)
  Nel 1815 una nave mercantile francese fa tappa all'Isola d'Elba. Una lettera di Napoleone in esilio viene consegnata al capitano della nave perché consegni a un uomo a Marsiglia. Prima di morire di...
● Il figlio di Montecristo (docId=2957, score=2,9863)
  Nel 1865 il generale proletario Gurko Lanen (George Sanders) diventa il dittatore dietro le quinte del Granducato di Lichtenburg situato nei Balcani. Gurko sopprime il clero e la libertà di stampa e i...
```

## 8. Query : trama esilio prigione francese e vendetta

Con questa query si verifica la capacità del sistema di restituire documenti rilevanti anche in presenza di più termini non sequenziali. È utile per testare l'efficacia dello **ItalianAnalyzer** nell'analisi morfologica del testo e nel riconoscimento delle parole derivate.

```
Query > trama esilio prigione francese e vendetta
Query interpretata: content:esil content:prigion content:frances content:vendett
● Trovati 512 documenti (ricerca in 236 ms):
● Il Conte di Montecristo (docId=1366, score=4,6987)
  Nel 1815 una nave mercantile francese fa tappa all'Isola d'Elba. Una lettera di Napoleone in esilio viene consegnata al capitano della nave perché consegni a un uomo a Marsiglia. Prima di morire di...
```

## 9. Query : trama "il fantastico sottomarino dell'enigmatico Capitano Nemo"

Questa query utilizza nuovamente una *PhraseQuery*, ma applicata al campo **content** (trama). Serve a verificare che la ricerca di sequenze di parole nel contenuto testuale funzioni correttamente e che l'analyser italiano gestisca in modo coerente la tokenizzazione e lo stemming.

```
Query > trama "il fantastico sottomarino dell'enigmatico Capitano Nemo"
Query interpretata: content:"? fantastico sottomarin enigmatic capitan nemo"
● Trovati 1 documenti (ricerca in 1 ms):
● 20.000 leghe sotto i mari (docId=388, score=11,2499)
  Una strana gigantesca "creatura marina" imperversa nei mari. La nave da guerra americana Abraham Lincoln viene inviata a indagare, ma viene speronata dalla "creatura" che si scopre essere il Nautilus,....
```

## 10. Query : trama (guerra OR battaglia) AND NOT fantascienza

Questa query è stata ideata per verificare il comportamento del parser nella gestione di operatori booleani (AND, OR, NOT). Permette di osservare come Lucene costruisca la query combinando più condizioni logiche e restituisca solo i documenti che rispettano i vincoli specificati.

```
Query > trama (guerra OR battaglia) AND NOT fantascienza
Query interpretata: +(content:guerr content:battagl) -content:fantascienz
● Trovati 336 documenti (ricerca in 3 ms):
● La casa con le persiane chiuse (docId=32, score=4,1638)
  Durante la Guerra Civile Americana un giovane soldato perde i nervi in battaglia e scappa a casa per nascondersi; sua sorella indossa la sua uniforme e prende il posto del fratello nella battaglia e...
● La Testa di Rame (docId=1869, score=3,9871)
  All'inizio della guerra civile americana, il presidente Abraham Lincoln chiede a Milt Shanks, proprietario di una fattoria nell'Illinois, di unirsi ai Copperheads, un'organizzazione quasi politica clandestina.
● Resa (docId=1767, score=3,9736)
  Durante la prima guerra mondiale, l'astuto e attraente sergente francese Dumaine, prigioniero di guerra, viene sequestrato in un campo di prigionieri al castello di un orgoglioso nobile e general...
● Civiltà (docId=828, score=3,9318)
  Il film si apre con lo scoppio di una guerra nel regno precedentemente pacifico di Wredpyrd. Il conte Ferdinando è l'inventore di un nuovo sottomarino a cui viene assegnato il comando della nuova nave...
```

I risultati ottenuti da queste dieci query confermano che il sistema gestisce correttamente la maggior parte delle casistiche comuni, incluse varianti morfologiche, accenti, numeri, frasi esatte e piccole imprecisioni di input, dimostrando un comportamento affidabile e coerente.

## Conclusioni

Il progetto ha dimostrato l'efficacia di **Apache Lucene** come motore di ricerca testuale, evidenziando prestazioni elevate sia in termini di velocità di indicizzazione che di accuratezza nella ricerca dei documenti. La pipeline realizzata, composta da una fase di **pre-processing in Python** e da un motore di ricerca sviluppato in **Java**, rappresenta un flusso completo per la costruzione di un sistema di **Information Retrieval**, che parte dalla preparazione del corpus testuale fino all'interrogazione interattiva dell'indice tramite console.

Le query di test eseguite hanno permesso di verificare l'affidabilità e la correttezza del sistema, mostrando come l'utilizzo di analyzer appropriati consenta di gestire efficacemente diverse casistiche, tra cui:

- variazioni morfologiche delle parole ricercate;
- differenze di maiuscole/minuscole e accenti;
- piccoli errori di battitura o inserimenti parzialmente errati;
- la presenza o assenza di stopwords nei titoli e nei testi.

Queste prove hanno evidenziato come la logica di indicizzazione e analisi implementata permetta di restituire in modo coerente e robusto i documenti più rilevanti rispetto alla query dell'utente, anche in presenza di input imperfetti o ambigui.