Homework 2

Indicizzazione e Ricerca con Apache Lucene

Irene Proietti Cesarini
Matricola: 562495

A.A. 2025/2026

Sommario

Il progetto propone un sistema completo per l'indicizzazione e la ricerca full-text di articoli scientifici, integrando strumenti di web scraping e tecniche di information retrieval.
A partire da un dataset di 50 articoli scaricati da arXiv, i documenti vengono convertiti in formato testuale, indicizzati con Apache Lucene e successivamente interrogati tramite un'interfaccia a linea di comando.

1 Introduzione

L'obiettivo del progetto è la realizzazione di un sistema in grado di indicizzare e ricercare in modo efficiente contenuti testuali derivanti da articoli scientifici. L'implementazione è suddivisa in tre fasi principali:

1. estrazione e conversione dei paper in formato `.txt`;
2. indicizzazione dei file mediante Apache Lucene;
3. ricerca ed analisi dei risultati tramite query testuali.

Il codice completo del progetto è disponibile al seguente indirizzo GitHub:

https://github.com/ireproces/hw2-idd.git

2 Tecnologie e Librerie Utilizzate

Il progetto è stato sviluppato in linguaggio **Java 21**, utilizzando il sistema di build Maven. Le principali dipendenze sono:

- **Apache Lucene 10.0.0**: per la creazione dell'indice e la gestione delle query;
- **lucene-analysis-common**: fornisce gli analyzer comuni;
- **lucene-queryparser**: consente il parsing e l'esecuzione delle query testuali.

Per la fase di estrazione dei dati è stato utilizzato uno script in **Python**, descritto nella sezione successiva.

3 Estrazione dei dati da arXiv

La directory `downloadfiles/papers_txt` contiene 50 articoli scientifici scaricati automaticamente dal portale *arXiv*. Lo script `donwload.py` gestisce il processo di scraping.

3.1 Download della pagina di ricerca

Nella prima fase, il programma utilizza la libreria `Selenium`, che consente di simulare la navigazione di un browser reale e di accedere ai contenuti dinamici generati dal sito.

Viene inizializzata un'istanza di browser (in questo caso, Chrome) e visitata la pagina dell'archivio contenente la lista dei paper su argomenti di Machine Learning. Lo script scorre automaticamente la pagina fino in fondo per garantire il caricamento di tutti i risultati, quindi salva il contenuto HTML completo in un file locale denominato `arxiv_page.html`.

Questo processo consente di acquisire in un'unica operazione l'elenco completo dei 50 paper restituiti dalla ricerca.

3.2 Parsing ed estrazione tramite XPath

Una volta salvata la pagina, il codice utilizza la libreria `lxml` per eseguire il parsing dell'HTML e costruire un albero DOM navigabile. Attraverso la funzione `html.fromstring()`, viene creata una struttura interna che può essere interrogata mediante espressioni **XPath**.

Ciascun paper è rappresentato nella pagina da un elemento `<li>` con classe `arxiv-result`. Lo script itera su tutti questi nodi per estrarre titolo e abstract, utilizzando le seguenti query XPath:

- per individuare il paragrafo contenente il **titolo** del paper:

  ```
  .//p[contains(@class, 'title')]/text()
  ```

- per catturare l'**abstract** completo, anche in presenza di nodi figli annidati:

  ```
  .//span[contains(@class, 'abstract-full')]/text() |
  .//span[contains(@class, 'abstract-full')]//text()
  ```

3.3 Generazione dei file testuali

Per ciascun risultato vengono poi creati file di testo separati contenenti le informazioni estratte. Ogni file include il titolo e l'abstract del paper nel formato:

```
Title: <titolo>
Abstract: <testo>
```

I documenti vengono salvati nella directory locale `papers_txt` e denominati secondo la convenzione `XXX_Title.txt`, dove:

- `XXX` è un identificativo numerico progressivo (da 001 a 050);
- `Title` deriva dal titolo del paper con gli spazi sostituiti da caratteri di sottolineatura.

4 Indicizzazione con Apache Lucene

La classe `TxtIndexer.java` implementa la logica del processo che consente di trasformare l'insieme dei file testuali generati nella fase precedente in un indice ottimizzato per la ricerca full-text.

4.1 Struttura dei documenti indicizzati

Per ogni file `.txt` presente nella directory dei paper, il programma crea un oggetto `Document` contenente quattro campi principali:

- **path**: percorso assoluto del file nel filesystem;
- **filename**: nome del file;

- **title**: titolo del paper;
- **abstract**: testo dell'abstract del paper.

I campi *path* e *filename* sono definiti come `StringField` e, pertanto, non vengono analizzati ma memorizzati così come appaiono, permettendo la ricerca esatta e la ricostruzione del file di origine.
I campi *title* e *abstract*, invece, sono di tipo `TextField` e vengono analizzati, consentendo ricerche full-text e matching parziali sui termini.

4.2 Analyzer utilizzati

Per la fase di tokenizzazione e analisi linguistica è stato impiegato lo **StandardAnalyzer**. Questo analyzer effettua:

- la suddivisione del testo in token (parole significative);
- la normalizzazione in minuscolo;
- la rimozione della punteggiatura e delle stop words più comuni.

La scelta dello `StandardAnalyzer` risulta adeguata poiché i contenuti dei paper di arXiv sono in lingua inglese e contengono terminologia tecnica coerente con il vocabolario gestito da tale analyzer.

4.3 Creazione e salvataggio dell'indice

L'indice risultante, salvato in una directory locale denominata `indexes_lucene`, contiene un documento per ciascun file analizzato, permettendo ricerche efficienti sia sui titoli sia sui contenuti testuali degli abstract.

Questa architettura garantisce una separazione logica tra metadati e testo, rendendo il sistema facilmente estendibile a ulteriori campi o analyzer specializzati.

5 Ricerca e query testuali

La ricerca è gestita dalla classe `TxtSearcher.java`, che utilizza le API di Apache Lucene per interrogare l'indice creato nella fase precedente.

In particolare, viene impiegato un oggetto `IndexSearcher` per eseguire le query, supportato da un `QueryParser` per l'interpretazione delle stesse.

5.1 Interfaccia di ricerca

L'interfaccia a linea di comando consente all'utente di inserire query in un formato controllato. Sono accettate esclusivamente le query che fanno riferimento ai campi indicizzati `title` e `abstract`, in una delle seguenti forme sintattiche:

- `title:term` – ricerca di un termine all'interno dei titoli;
- `abstract:term` – ricerca di un termine nei testi degli abstract;
- `abstract:"phrase"` – ricerca di una frase esatta nei testi degli abstract;
- `title:x,abstract:y` – esecuzione combinata di due ricerche, una per il titolo e una per l'abstract.

In tutti i casi, il testo della query viene analizzato tramite lo StandardAnalyzer, lo stesso utilizzato in fase di indicizzazione, assicurando coerenza nella tokenizzazione e nel filtraggio dei termini.
Il parser Lucene traduce la query testuale in una struttura interna che il motore di ricerca può eseguire in modo efficiente.

5.2 Esecuzione della ricerca

Una volta interpretata la query, il sistema invoca il metodo `search()` dell'oggetto `IndexSearcher`, ottenendo una lista ordinata di risultati (`TopDocs`).
Per ciascun documento recuperato, vengono stampate a video le informazioni principali:

- il titolo del paper;
- il testo del suo abstract;
- lo score di rilevanza (calcolato tramite la funzione BM25) indicativo del grado di corrispondenza rispetto alla query.

6 Risultati e statistiche

6.1 Tempi di indicizzazione

Il tempo medio di indicizzazione dell'intero dataset (composto da 50 documenti) è stato di circa **0,52 secondi**. Questi risultati dimostrano l'efficienza del motore Lucene anche su dataset di dimensioni ridotte, con tempi di risposta praticamente istantanei.

6.2 Esempi di query e tempi di ricerca

Durante la fase di validazione del sistema sono state eseguite diverse query rappresentative, volte a verificare la correttezza dell'indicizzazione, la qualità del ranking e la robustezza del parser nei confronti di input non standard.
Tra le query più significative si riportano:

- `title:diffusion` – ricerca di un termine tecnico all'interno dei titoli, con più risultati pertinenti;
- `abstract:"neural network"` – ricerca di una frase esatta all'interno degli abstract;
- `abstract:transformer` – termine frequente, che restituisce numerosi risultati con score differenziato;
- `title:graph,abstract:learning` – query combinata su due campi distinti, utile per verificare il corretto parsing e la fusione dei risultati.

In tutti i casi, i risultati restituiti da Lucene sono stati ordinati coerentemente con la pertinenza dei contenuti, mostrando tempi di risposta medi inferiori ai 300 millisecondi anche su dataset di 50 documenti.

6.3 Casi limite e query non valide

Per valutare la robustezza sono state testate anche alcune query non corrette o prive di corrispondenze, come:

- `author:smith` – campo non indicizzato;
- `abstract:aaaaa` – termine inesistente, non restituisce alcun risultato;
- `title:"deep learning for graphs"` – forma sintattica non riconosciuta.

In tutti i casi, il sistema ha reagito correttamente: le query non valide vengono intercettate e segnalate dal parser, mentre quelle senza risultati vengono gestite con messaggi espliciti, senza errori di esecuzione o interruzioni del programma.

```
Query > title:diffusion

-----Documents-----
Found 1 documents.
```

Figura 1: Output parziale della query title:diffusion

```
Query > abstract:"neural network"

-----Documents-----
Found 5 documents.
```

Figura 2: Output parziale della query abstract:"neural network"

```
Query > title:graph,abstract:learning

No documents found for the query.

-----Fields queried-----
• Title queried: true
• Abstract queried: true

-----Search Statistics-----
• Total documents returned: 0
• Total time: 0,02 seconds
```

Figura 3: Output completo della query title:graph,abstract:learning

7 Conclusioni

Il progetto ha mostrato come sia possibile costruire un sistema di ricerca full-text completo utilizzando strumenti open-source. Lucene si è rivelato particolarmente potente e flessibile, permettendo di ottenere un motore di ricerca performante e facilmente estendibile.
Alcuni possibili sviluppi futuri includono:

- introduzione di un'interfaccia grafica o web;
- supporto per query booleane complesse;
- integrazione di un analyzer multilingua o customizzati.