DBLP
full-text search system

Gestione dell'informazione 2018/2019.

Perchè Lucene ?

Lucene è un progetto free ed open source implementato in Java, considerato lo standard de-facto per le ricerche di tipo commerciale. Questa è la libreria per “Information Retrival” più utilizzata nell’industria.

Utilizza diversi modelli di ranking tra cui: BM25Similarity Model e Boolean Model.

Per citare alcuni colossi, che la sfruttano per le loro ricerche abbiamo Amazon (Search Inside This Book), Netflix, Myspace, LinkedIn, FedEx e molti altri.

Problemi incontrati

Parsing ed indicizzazione corretta del file con suddivisione delle pubblicazioni (considerando le loro categorizzazioni) e delle venue

Riscrittura della query utente in modo da soddisfare la sintassi proposta nelle richieste del progetto ed allo stesso tempo rendere il testo cercato interpretabile dai queryParser di Lucene.

Gestione dei risultati e del loro ordine anche in casi in cui vi siano query particolari
Esempio >> publication.author: Vianu AND venue: PODS

Stampa dei risultati permettendo all'utente di interagire e nello stesso tempo dargli un feedback utile.

Searched Words highlighting: evidenziare all'interno dei risultati proposti all'utente le parole da lui utilizzate per effettuare la ricerca.

Parsing ed Indicizzazione

Il primo problema che ci siamo ritrovati a dover affrontare è stato quello del parsing e della indicizzazione. L'elevata richiesta di entità combinata ai 2,5GB del file .xml comportavano elevatissimi costi in termini di tempo. Mediante l'introduzione dei Thread, in punti strategici della funzione rebuildIndexes(), siamo riusciti ad ovviare a tale problema.

Indicizzazione: nello specifico

Per fare in modo che l'utente potesse essere in grado di indicizzare un nuovo documento in un tempo relativamente breve era necessario ottimizzare non tanto il parsing ma quanto l'indicizzazione.

Da qui la scelta della sulla suddivisione, di questo processo, in Thread. Nella sua prima versione, però, comportava la messa in parallelo di ogni istanza del documento.
Questa strategia, purtroppo, non era ottimale: la presenza di tutte queste istanze esauriva rapidamente la memoria ed allo stesso tempo i thread entravano in conflitto fra loro nella fase della scrittura su file.

Indicizzazione: nello specifico

Si è pensato quindi di eseguire la funzione di indicizzazione ***indexArticle()*** in maniera lineare ed il suo valore di ritorno, invece, viene gestito parallelamente dai diversi thread. Per evitare il conflitto tra thread è stata aggiunta una sincronizzazione, che permette ad essi di accedere in contemporanea ai file di indice e di poter accedere in maniera non concorrente, in scrittura, al file.

Riscrittura delle query

Per supportare la sintassi proposta dalle specifiche del progetto si è reso necessario effettuare una riscrittura delle query, questo perchè la sintassi richiesta e quella del queryParser di Lucene differiscono sotto alcuni aspetti.

Abbiamo quindi creato due funzioni all'interno della classe ***SearchEngine,*** che si occupa della gestione dell'esecuzione e gestione delle query, per la manipolazione della stringa contenente la query dell'utente.

Le funzioni per la riscrittura sono: ***rewriteQuery()*** e ***splittedQuery()***

Riscrittura delle query: nello specifico

La funzione ***rewriteQuery()*** si occupa di adattare la query alla gerarchia degli elementi che compongono il DBLP:

>> ***field***: *pub-search | venue-search*
pub-search *: pub-ele[.pub-field]*
pub-ele: *publication | article | incollection | inproc | phThesis | masterThesis*
pub-field: *author | title | year*
venue-search: *venue-ele[.venue-field]*
venue_ele: *book | proceedings*
venue-field: *title | publisher*

Permettendoci così di gestire gli elementi sui quali effettuare la ricerca.

Riscrittura delle query: nello specifico

La funzione ***splittedQuery()*** ha invece il compito di gestire i casi in cui la query è eseguita su pubblicazioni e venue ad esempio
Esempio >> publication.author: Vianu AND venue: VLDB

In questo caso la funzione viene utilizzata per splittare la query in due indipendenti tra di loro in particolare una da eseguire sulle pubblicazioni ed una sulle venue.

Successivamente le query ottenute dalla manipolazione di quella principale verranno gestite da altre funzioni mostrate a seguire, con lo scopo di trovare i match tra gli elementi che le soddisfano ed ordinarli per pertinenza.

Gestione dei risultati

Il progetto prevede che nel caso in cui l'utente effettui una query del tipo
>> publication.author: Vianu AND venue: VLDB vengano mostrati all'utente in ordine di pertinenza le pubblicazioni con autore Vianu presenti nelle conferenze relative a VLDB.

Per questo si è reso necessario implementare diverse funzioni, le più rilevanti tra tutte sono:
> ***createCrossreffedPubList()***
> ***findMetchingResults()***
> ***ordinaMatchingResults()***

Gestione dei risultati: nello specifico

La funzione ***createCrossreffedPubList()*** crea una lista contenente le pubblicazioni che soddisfano la query dell'utente e sono connesse a delle venue attraverso un crossref.

A questo punto ***findMatchingResults()*** recupera questi elementi e li memorizza eliminando le ridondanze in una struttura apposita. In seguito ordina la struttura utilizzando la funzione ***ordinaMatchingResults().***

Ora abbiamo a disposizione tutti gli elementi per comporre i risultati da mostrare all'utente, in ordine di pertinenza, che verranno riuniti in una struttura unica (***allDocs***) generata prima della stampa.

Stampa dei risultati

I risultati sono ora ordinati e raccolti nella struttura (***allDocs***) come descritto precedentemente.

Attraverso le funzioni ***stampaRisultati()***, ***stampaRisultato()*** e ***stampaCampoRisultato()*** verrà quindi generato un output strutturato con il quale l'utente potrà interagire.

L'interazione prevede di consultare i risultati attraverso lo scorrimento di pagine e la possibilità, nel caso di query su venue e pubblicazioni, di visualizzare la venue relative ad una certa pubblicazione, semplicemente cliccando sul campo *Vedi anche:* di quest'ultima.

Stampa dei risultati: nello specifico

stampaRisultato(): in caso di query indicanti venue, la funzione di stampa stessa, andrà a generare un link. Questo sarà collegato ai crossref relativi alla pubblicazione

Esempio>>publication.author:Vianu and venue:VLDB

Al click viene quindi stampata la venue (come nel DBLP), a questa collegata, nel JTextPane cancellando i precedenti risultati. Potremo recuperare tali risultati clickando su *Torna ai risultati*

Searched Words highlighting

Nella realizzazione della parte grafica abbiamo pensato di rendere più facilmente consultabile la parte relativa alla consultazione dei risultati, generati dalle ricerche eseguite utente.

Abbiamo quindi ritenuto opportuno riprogettare la funzione di stampa, dandole uno stile ben delineato, la possibilità di poter consultare i crossref e messo in risalto le parole chiave cercate nella query.

Searched Words highlighting: nello specifico

splittedQuery(): funzione che si occupa dell'estrazione, mediante splitting, delle informazioni a noi necessarie per la stampa. Essa restituirà una stringa con la parte della query relativa alle pubblicazioni, seconda parte di questa relativa alle venue ed, infine, una stringa con tutte le parole contenute nella query separate da uno spazio. Tale stringa verrà passata alla funzione ***stampaRisultato()*** che si occuperà di estrapolare le parole chiave ed inserirle in un albero che ci permetterà, successivamente, di dare uno stile ben definito e di facile lettura ai risultati richiesti dall'utente.

Searched Words highlighting: nello specifico

stampaRisultato(): come già anticipato nella slide precedente, non appena verrà invocata, la funzione inserirà queste parole in un albero e procederà con la loro ricerca all'interno dei risultati ottenuti. Infine, attraverso l'uso di un JTextPane, potremo dare uno stile e quindi andare ad evidenziare, se presenti, le parole chiave inserite dall'utente.

Tipologia di Query supportate

Fin da subito si è pensato di supportare la maggiorparte delle query possibile:

- Singolo termine	Lucene
- Frase	“Lucene è una libreria esterna”
- Campi	nomeCampo:”Lucene è una libreria esterna”
- Ricerche wildcard	Lu?ene, Lucen*
- Ricerche fuzzy	Luc~
- Ricerche di prossimità	“Lucene è”~10
- Boosting term	Lucene^2 libreria

Grazie per l’attenzione

Fabio, Simone e Valentino.