Strutture indice

Inverted file

Lo scopo delle strutture indice in un IRS è quello di incrementare l'efficienza del sistema. Le strutture a indice consentono di:

1. velocizzare le operazioni di ricerca;
2. ottimizzare l'accesso alle informazioni dei termini della collezione;
3. ottimizzare le ricerche dei singoli termini.

La tabella delle occorrenze come prodotta nel formato documento/termine descritto durante le lezioni precedenti è *fortemente sparsa*. Questo perché molte delle celle della matrice vengono lasciate vuote al fine di indicare la non occorrenza di un termine all'interno di un particolare documento. Il risultato di questa scelta è che la memorizzazione della matrice in memoria richiede una quantità di spazio molto maggiore a quella necessaria altrimenti. Un altro effetto indesiderato di questa soluzione è che ottenere l'elenco dei documenti che contengono uno specifico termine risulta particolarmente costoso in quanto è necessario scorre la lista dei termini associata a ciascun documento.

Per far fronte a questi problemi si adotta una soluzione differente. La tabella delle occorrenze viene *invertita* e quindi ad ogni termine vengono associati i documenti in cui questo occorre. Questa inversione rende l'accesso all'elenco dei documenti contenenti uno specifico termine, una delle operazioni fondamentali per un IRS, molto più veloce. Questo tipo di struttura dati è nota con il nome di struttura a *File Inverted* e può essere utilizzata indipendentemente dal modello di retrieval adottato dal sistema.

Dizionario e posting file

In una struttura dati a File Inverted i termini indice della collezione vengono mantenuti all'interno di un file detto *dizionario*. Nel dizionario di una collezione, ogni termine viene memorizzato insieme alla sua frequenza globale, ossia al suo numero di occorrenze all'interno della collezione. Questo facilita il calcolo l'IDF e ottimizza la valutazione di query booleane.

Come affermato in precedenza, ad ognuno dei termini della collezione, ossia quelli presenti all'interno del dizionario, viene associata la lista dei documenti in cui questo è contenuto. Questa lista viene memorizzata all'interno di un file chiamato *posting file*.

Per ogni documento puntato viene quindi indicato l'identificatore univoco del documento, o *DocID*, e vengono memorizzate la frequenza del termine nel documento, denotata come *tf* e la posizione di ogni occorrenza del termine. Questa seconda informazione è opzionale, in quanto utile solamente per sistemi che consentono la formulazione di query con contesto*, e può essere espressa in termini di numero di parole o byte dall'inizio del documento, oppure indicando il numero di sezione, paragrafo, frase e parola in

cui il termine è presente all'interno del documento. Questo consente la valutazione di operatori di adiacenza e di vicinanza e la progettazione di interfacce utente per l'evidenziazione delle occorrenze dei termini ricercati all'interno dei documenti reperiti.

* le query con contesto sono le query che utilizzano operatori di adiacenza e simili.

In termini di occupazione di memoria, lo spazio richiesto per la memorizzazione del dizionario è piuttosto ridotto. Il vocabolario viene infatti memorizzato in maniera piuttosto efficiente e cresce secondo la legge di Heap, ossia come $O(n^B)$, dove n è lo spazio occupato dal testo dei documenti B è una costante tra 0.4 e 0.6. La memorizzazione delle occorrenze dei termini richiede invece molto più spazio. Questo perché ogni parola che compare nel testo è riferita una volta nella struttura dati e quindi lo spazio cresce come $O(n)$.

Tecniche di ottimizzazione

Ai fini di ottimizzare il posting file si applicano tecniche di ottimizzazione alle diverse componenti di questo: i nomi dei termini, gli identificatori dei documenti, il conteggio delle occorrenze dei termini e la posizione di questi all'interno del documento. Le principali tecniche di ottimizzazione utilizzate sono:

- *compressione dei DocID con indirizzo relativo*, in cui la lista dei DocID viene ordinata e ad ogni DocID d_i viene sostituita la distanza $d_i - d_{i-1}$; si tratta di una tecnica molto semplice ma che porta ad una riduzione delle dimensioni dell'indice del 10-15%;
- *divisione a blocchi del testo del documento*, in cui il testo di ogni documento viene diviso in blocchi e ogni occorrenza punta ai blocchi in cui compare la parola. La riduzione dell'occupazione di memoria prodotta dall'utilizzo di questa tecnica è legata sia al fatto che la dimensione del puntatore al blocco è minore rispetto a quella dell'offset rispetto all'inizio del documento, sia al fatto che tutte le occorrenze di una parola contenute all'interno di un singolo blocco utilizzeranno lo stesso riferimento. Lo principale svantaggio causato da questa tecnica è rappresentato dal fatto che le ricerche per contesto richiederanno più operazioni. Questo perchè un volta identificato che due termini occorrono all'interno dello stesso blocco sarà necessario la lettura del blocco e lo scorrimento* di questo al fine di verificare l'adiacenza.

* scusate se scorro.

In un IRS che adotta una struttura a File Inverted, a fronte di una ricerca devono essere eseguite le seguenti *operazioni*:

- accesso al file dizionario e ricerca dei termini nella query;
- reperimento della lista di posting associata ad ognuno dei termini trovati all'interno del dizionario;
- filtraggio dei risultati, questo perché, se la query contiene più termini connessi da operatori, le liste parziali dei risultati andranno fuse in un'unica lista globale.

Inoltre, come sottolineato in precedenza, se la query utilizza operatori di prossimità o adiacenza e il File Inverted adotta un indirizzamento a blocchi, è necessario scandire ognuno dei blocchi contenenti i termini della query in maniera tale da ottenere gli offset delle occorrenze dei termini.

Nel caso di File Inverted che utilizzano un *indirizzamento lineare* e non a blocchi, l'indice può essere acceduto molto velocemente, ad esempio, utilizzando una ricerca binaria in $O(logn)$. Inoltre, si ha un miglioramento delle performance di valutazioni sequenziali e un risparmio in termini di occupazione di memoria. Il problema fondamentale di questa tecnica si evidenzia nel caso di collezioni il cui contenuto può cambiare (possono essere aggiunti o tolti documenti), in caso di modifica del contenuto infatti gli indici costruiti utilizzando un indirizzamento verranno invalidati e dovranno essere ricostruiti.

Indice a B-Tree

Un'altra strategia di indirizzamento degli indici all'interno di una struttura ad Inverted File è quella basata su *B-Tree*. Un indice a B-Tree di ordine d utilizza una struttura ad albero bilanciato in cui la radice contiene da 1 a d termini e ogni altro nodo contiene da $d/2$ termini a d termini. L'idea alla base del funzionamento di questo tipo di struttura è che se k_i è l'i-esimo termine in un nodo intermedio:

- tutti i termini nell'$i-1$-esimo figlio sono lessicograficamente minori di k_i;
- tutti i termini nell'i-esimo figlio sono lessicograficamente maggiori di k_i.

Questa strategia porta dei significativi miglioramenti rispetto ai problemi segnalati per le strategie precedenti. Strutture di questo tipo possono infatti essere accedute molto velocemente: $O(log_d n)$, dove d e n sono ordine e profondità dell'albero. Oltre ai vantaggi in termini di tempo, anche dal punto di vista dello spazio si evidenziano dei miglioramenti.

In alcuni contesti questa strategia può però risultare inadatta, in particolare:

- in presenza di molte ricerche sequenziali, le ricerche sequenziali in questo contesto risultano infatti particolarmente inefficienti;
- se l'indice viene memorizzato su disco, questo perché ogni nodo può richiedere un accesso al disco separato;
- nel caso in cui si effettuino molte operazioni di inserimento, questo perché nel caso peggiore l'albero risultante potrebbe essere completamente sbilanciato.

Indice a Suffix-Tree

La strategia di indirizzamento a *Suffix-Tree* tratta il documento come se fosse un unica stringa e lavora basandosi sui diversi suffissi di questa stringa. Ognuno di questi suffissi viene identificato utilizzando il suo indice iniziale. Dal testo vengono selezionati dei punti indice che possono essere reperiti e i puntatori ai suffissi vengono memorizzati nelle foglie dell'albero prodotto, in maniera tale da memorizzare una sola volta le parti comuni tra i diversi suffissi.

Questa strategia è particolarmente adatta a ricerche complesse, sia ricerche di intere frasi che ricerche sequenziali di termini o prefissi. Gli svantaggi legati all'utilizzo di questa tecnica sono rappresentati dall'elevato costo del processo di creazione dell'indice e dalla maggiore occupazione di memoria rispetto alla tecniche precedenti (120-240% della dimensione della collezione).

Indice a Signature

L'ultima strategia di indirizzamento che abbiamo visto è quella *a Signature*. Una strategia di indirizzamento a Signature definisce una funzione $h(x)$ che mappa gli indici x in maschere di B bit dette signature.

Ad ogni blocco viene associata una *signature*, una maschera di bit. La signature di un blocco viene prodotta effettuando l'OR delle stringhe di bit associate dalla funzione h a ciascuno dei termini presenti nel blocco.

L'*indice signature* di un documento è costituito da una sequenza sequenza di coppie (signature, puntatore al blocco), una per blocco appartenente al documento.

In fase di ricerca di un documento, viene confrontata la signature prodotta dalla funzione h per il termine ricercato (query) con quelle facenti parte dell'indice di ognuno dei documenti. Nel caso in cui la signature della query sia contentuta in almeno una delle signature presenti nell'indice di un documento, il documento viene reperito.

L'applicazione di questo tipo di tecnica di indicizzazione è prevalente per collezioni di dati multimediali e testi non troppo grossi grazie alla ridotta occupazione di memoria che la contraddistingue (10-20% rispetto alla dimensione della collezione).

Gli svantaggi di questa tecnica sono rappresentati dalla problematicità delle ricerche sequenziali e dall'eventualità di false drop.

False drop

Un *false drop* è la probabilità che una signature di un blocco soddisfi una query anche se il blocco non contiene il termine ricercato. Questo avviene molto spesso nel caso di signature con troppi bit a 1.

La scelta di indicizzare a blocchi i documenti è legata a questo problema, si osserva infatti che i false drop si minimizzando andando a considerare parti di documento più piccole, la cui firma conterrà probabilmente un numero ridotto di 1. Infatti, se un blocco è più piccolo significa che contiene meno termini; se contiene meno termini significa che la sua firma è il prodotto di un numero di OR ridotto e quindi la probabilità di bit a 1 è più bassa.