Università degli studi di Padova

Facoltà di Ingegneria

Tesi di Laurea Specialistica in

Ingegneria Informatica

Titolo
LSPR: un modello di reperimento dell'informazione

Relatore

Prof. Massimo Melucci

Candidato

Alberto Costa

Anno Accademico 2008/2009

Università degli studi di Padova

Facoltà di Ingegneria

Tesi di Laurea Specialistica in

Ingegneria Informatica

Titolo
LSPR: un modello di reperimento dell'informazione

Relatore
Prof. Massimo Melucci

Candidato
Alberto Costa

Anno Accademico 2008/2009

Sommario

Lo scopo di questa tesi è di analizzare, implementare e valutare le prestazioni di un nuovo modello di reperimento dell'informazione basato su DFT (*Discrete Fourier Transform*).

I modelli principali su cui si basano i motori di ricerca attuali sono 3: booleano, vettoriale e probabilistico (e loro varianti).

Ciò che viene proposto con questo lavoro è però qualcosa di nuovo, e non una variante di un modello già esistente. Infatti, in letteratura non esistono motori di ricerca basati su DFT che abbiano come target i file di testo.

L'idea che sta alla base di questo modello è di associare a ogni termine dell'interrogazione un segnale sinusoidale, avente una ampiezza e una frequenza determinate in maniera opportuna, in modo da poter pensare la query non più come un insieme di termini, ma come la somma dei segnali sinusoidali associati ai termini stessi. L'interrogazione potrà così essere analizzata nel dominio delle frequenze (e qui interviene la DFT), ottenendo così uno spettro. Per ogni documento della collezione, che nel modello corrisponde a un filtro, lo spettro associato alla query viene filtrato.

Il ranking è legato alla potenza dello spettro dell'interrogazione dopo l'operazione di filtraggio: più un documento riesce a ridurre la potenza dello spettro dell'interrogazione, più il suo rank sarà elevato.

Abstract

The aim of this thesis is to analyze, implement and test a new model of information retrieval based on DFT (*Discrete Fourier Transform*).

Currently, search engines are based on 3 principal models: boolean, vector and probabilistic (including their variants).

What is suggested with this work is something new, and not the variant of a model that already exists. As a matter of fact, search engines based on DFT that shoot for text files do not exist in literature.

The idea at the base of this model is to associate to each term of the query a sine curve, with both amplitude and frequency set in an opportune way, so the query move from a set of terms to the sum of sine curves related to the terms. The query can be analyzed in this way in the frequency domain (and here DFT is used) in order to obtain a spectrum. For each document of the collection, which corresponds to a filter in the model, query's spectrum is filtered.

Ranking is related with the power of query's spectrum after filtering: the more the document is able to decrease the power of query's spectrum, the more its rank will increase.

Notazione

- j : rappresenta il valore $\sqrt{-1}$, quando non è specificato diversamente;
- A^* : sia $A \in \mathbb{C}$. A^* rappresenta il complesso coniugato di A. In altre parole, se $A = x + jy$, allora $A^* = x - jy, \quad x, y \in \mathbb{R}$;
- $x[i]$, $1 \leq i \leq |x|$: rappresenta l'elemento i-esimo del vettore x; si noti che gli indici partono da 1, anche se solitamente nei linguaggi di programmazione si parte da 0. Per le matrici la notazione è simile (se T è una matrice, $T[i, j]$ rappresenta l'elemento posto alla i-esima riga e alla j-esima colonna);
- x_{ij} : rappresenta il termine i-esimo nel documento j-esimo;
- X_j : rappresenta il documento j-esimo; se tale documento è composto da m termini scriveremo $X_j = \{x_{1j}, x_{2j}, \ldots, x_{mj}\}$;
- Q : rappresenta l'interrogazione (query), esclusi quei termini che non appaiono nella collezione; in modo simile ai documenti, se l'interrogazione è composta da t termini scriveremo $Q = \{q_1, q_2, \ldots, q_t\}$, dove q_i è il termine i-esimo dell'interrogazione Q;
- $|X_j|$: rappresenta la cardinalità del documento j-esimo, intesa come il numero di termini di cui è composto. Nel caso di $X_j = \{x_{1j}, x_{2j}, \ldots, x_{mj}\}$, si avrebbe che $|X_j| = m$. Una considerazione analoga vale anche per le interrogazioni;
- C : rappresenta la collezione di documenti; se la collezione è composta da l documenti, scriveremo $C = \{X_1, X_2, \ldots, X_l\}$;
- $|C|$: rappresenta la cardinalità della collezione; è equivalente a l, nel caso precedente;
- w_{ij} : rappresenta il peso del termine i-esimo nel documento j-esimo; la versione normalizzata si indica con $\hat{\mathrm{w}}_{ij}$;

- N : rappresenta il numero di campioni del segnale a tempo discreto. È anche uguale al numero di campioni usati per il calcolo della DFT, sebbene si possano usare più campioni per aumentare l'accuratezza;
- F : quanto di frequenza, ovvero distanza tra due campioni successivi della trasformata di un segnale a tempo discreto;
- T : quanto di tempo, ovvero distanza tra due campioni successivi di un segnale a tempo discreto;
- F_c : rappresenta la frequenza di campionamento; si ha che $F_c = N \cdot F = \frac{1}{T}$;
- T_p : rappresenta il periodo del segnale a tempo discreto. Si ha la relazione $T_p = N \cdot T = \frac{1}{F}$;
- Supp$(A \rightarrow B)$: indica il supporto della regola associativa $(A \rightarrow B)$;
- Conf$(A \rightarrow B)$: indica la confidenza della regola associativa $(A \rightarrow B)$.

Indice

Capitolo 1

Presentazione

Nell'ambito del reperimento dell'informazione (o IR), uno degli aspetti più importanti è la definizione dei cosiddetti *modelli di reperimento dell'informazione*. Una definizione di modello di IR, riportata in [1], è la seguente:

> Un modello di reperimento dell'informazione è un insieme di costrutti che sono formalizzati allo scopo di rendere possibile la rappresentazione del contenuto di documenti e interrogazioni, nonchè di definire l'algoritmo di reperimento dei documenti in risposta ad un'interrogazione.

Il modo più semplice per descrivere un modello di reperimento dell'informazione è quello di usare una metafora, che in questo contesto serve per spiegare a livello intuitivo come il modello funziona. I modelli principali, come quello booleano, vettoriale e probabilistico, hanno tutti una loro metafora; anche il modello *LSPR* ne ha una, e verrà ora definita, usando i simboli introdotti nella notazione per evidenziare meglio la relazione tra la metafora e il modello.

Supponiamo che una persona debba recuperare un oggetto, e che per raggiungerlo abbia a disposizione un percorso Q. Tale percorso è però attraversato da radiazioni, diverse sia per l'intensità che per la frequenza. In particolare, indicando con $|Q|$ il numero di radiazioni diverse che attraversano questo percorso, possiamo scrivere $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$. Tale scrittura ha lo scopo di caratterizzare il percorso con l'insieme delle radiazioni che lo attraversa, e quindi q_i è la i-esima radiazione del percorso Q.

Come già accennato, ogni radiazione è definita da una frequenza e una intensità; per la radiazione q_i, questi parametri saranno chiamati rispettivamente f_i e A_i.

Supponiamo inoltre che la persona che deve recuperare l'oggetto abbia a disposizione $|C|$ tute protettive contro le radiazioni, e chiamiamo X_j la tuta j-esima. Esse sono caratterizzate dalla presenza di moduli che riescono

a bloccare, in maniera diversa, le radiazioni associate a certe frequenze. In particolare, si supponga che la tuta X_j contiene $|X_j|$ moduli: possiamo allora caratterizzarla come $X_j = \{x_{1j}, x_{2j}, \ldots, x_{|X_j|j}\}$, dove ogni modulo x_{ij} è definito da una frequenza f_i e da una capacità di bloccare radiazioni $\hat{\mathrm{w}}_{ij}$.

Un altra caratteristica della tuta è che sebbene riesca a bloccare completamente solo $|X_j|$ tipologie di radiazione, riesce comunque ad attenuare radiazioni con frequenze vicine a quelle che blocca. In altre parole, se la tuta blocca una radiazione a frequenza f_t, cioè ha un modulo x_{tj} caratterizzato da una frequenza f_t, riesce comunque ad attenuare una radiazione a frequenza f_s, se $|f_s - f_t| \leq \epsilon(\hat{\mathrm{w}}_{it})$.[1]

Lo scopo della persona che deve recuperare l'oggetto è allora quello di scegliere la tuta più "adatta" al percorso, cioè quella che blocca meglio le radiazioni, per massimizzare così la probabilità di non subire danni nell'attraversamento del percorso stesso.

A questo punto il parallelo tra la metafora e il modello di reperimento dell'informazione dovrebbe cominciare ad essere chiaro. Le tute rappresentano i documenti della collezione, mentre i vari moduli che le compongono sono i termini dei documenti. Ogni termine viene così caratterizzato da una frequenza e una capacità di bloccare la radiazione, e quest'ultima è il peso del termine nel documento.[2]

Il percorso è invece l'interrogazione, e le varie radiazioni q_i sono i suoi termini. L'operazione di scegliere le tute più promettenti, cioè i documenti più adatti all'interrogazione, è in pratica il ranking.

Resta però un dettaglio da definire: come si fa a decidere, avendo le informazioni su una tuta X_j e sul percorso Q, quanto questa tuta è sicura? In altre parole, come si decide quanto un documento è buono rispetto all'interrogazione? Per fare questo si usa la DFT. Infatti, una volta che si conoscono i valori di intensità e frequenza associati ai vari termini dell'interrogazione, essi sono usati, come già accennato, per definire dei segnali sinusoidali, in modo che al termine q_i, caratterizzato da intensità A_i e frequenza f_i, corrisponda il segnale $A_i \sin(2\pi f_i n T)$. Il modulo della DFT di tale segnale assume valore massimo, e dipendente da A_i, intorno a f_i, per poi diminuire gradualmente (se si presenta il fenomeno della dispersione spettrale, come spiegato nella sezione 3.1.3). In questo modo ogni interrogazione avrà uno spettro associato, con dei picchi (di ampiezza proporzionale ai valori A_i) vicino alle frequenze f_i dei termini che la compongono.

[1] Questo significa che la dimensione dell'intorno entro cui il modulo x_{tj} riesce ad attenuare le radiazioni dipende dalla sua capacità di attenuazione $\hat{\mathrm{w}}_{tj}$; in particolare, all'aumentare di $\hat{\mathrm{w}}_{tj}$, aumenta l'intorno.

[2] In realtà si vedrà in seguito che ogni termine è caratterizzato da un insieme di frequenze, infatti uno stesso termine può assumere due diverse frequenze in relazione a due diversi termini dell'interrogazione; questo però non modifica l'idea di base, e si è quindi preferito per semplicità considerare una sola frequenza nella metafora.

Alle frequenze definite dai vari termini x_{ij} dei documenti, si può pensare di avere dei filtri *notch*, cioè degli attenuatori, le cui caratteristiche dipendono da $\hat{w}_{ij}$.

Per sapere quanto un documento è buono rispetto all'interrogazione, si può allora filtrare lo spettro dell'interrogazione con i filtri notch ricavati dal documento, e fatto questo si calcola la potenza.[3]

Ovviamente, se l'interrogazione dopo il filtraggio dovuto al documento X_i presenta una potenza maggiore rispetto al filtraggio del documento X_j, questo significa che il secondo è più buono del primo rispetto all'interrogazione, o tornando alla metafora, significa che le radiazioni lungo il percorso Q sono attenuate meno dalla tuta X_i che dalla tuta X_j, per cui è da preferire la seconda alla prima. Nella fase di ranking troveremo quindi prima il documento X_j e poi X_i, ed è proprio dal fatto che il ranking inserisce prima i documenti il cui filtraggio produce una potenza minore che deriva il nome del modello: *LSPR* è infatti l'acronimo di *Least Spectral Power Ranking*.

Per quanto concerne la struttura della tesi, nel capitolo 2 verrà brevemente analizzato lo stato dell'arte rispetto ai modelli di reperimento dell'informazione; nel capitolo 3 si presenteranno le nozioni teoriche necessarie per comprendere le varie componenti del modello. Nel capitolo 4 si trova la spiegazione dettagliata del funzionamento di *LSPR* e il suo confronto con un altro modello usato come riferimento. Nel capitolo 5 verranno discusse le parti più importanti relative all'implementazione in linguaggio Java dei modelli sviluppati, e dopo la fase di test (con la collezione sperimentale CACM) presentata nel capitolo 6, si trovano le conclusioni, nel capitolo 7.

[3] In questo lavoro con potenza si intende la somma dei moduli di mezzo spettro, data la simmetria, anche se per il teorema di Parseval (il cui enunciato si può trovare in [5]) la vera potenza sarebbe proporzionale alla somma dei moduli al quadrato dell'intero spettro.

Capitolo 2

Stato dell'arte

I modelli di reperimento dell'informazione più importanti proposti in letteratura sono 3:

- modello booleano, negli anni 50,
- modello vettoriale, alla fine degli anni 60,
- modello probabilistico, negli anni 70.

Nel modello booleano, che trova la sua applicazione anche nei motori di ricerca per il Web, le interrogazioni sono operazioni logiche (come OR o AND) tra insiemi di documenti, e ciò che viene restituito sono i documenti che rendono vera l'interrogazione.

Il modello vettoriale invece considera i termini, i documenti e l'interrogazione, come dei vettori generati da una certa base. Un documento è giudicato dal modello tanto più rilevante quanto più sono vicini i vettori che rappresentano l'interrogazione e il documento stesso.

Nel modello probabilistico infine i documenti sono variabili aleatorie, così come la rilevanza, i cui valori rappresentano la probabilità di un documento di essere rilevante per una interrogazione.

Oltre a questi modelli ne esistono altri, proposti in seguito, i più importanti dei quali sono:

- modello statistico della lingua, verso la fine degli anni 90,
- modello basato su reti ipermediali, tra gli anni 80 e 90,
- modello di analisi della semantica latente, alla fine degli anni 80.

Nel modello statistico della lingua si vuole stimare la probabilità che interrogazione e documento siano stati generati dallo stesso modello di lingua, essendo quest'ultimo una distribuzione di probabilità valida per tutte le frasi della lingua.

Il modello basato su reti ipermediali cerca invece di strutturare, attraverso un ipertesto, una collezione di documenti come una rete che connette i documenti stessi. I collegamenti possono essere di diversa natura, e definire tra i documenti relazioni di tipo statistico (basate sulla similarità), semantico (ad esempio collegate a bibliografia o citazioni) e altre.

Il modello di analisi della semantica latente (LSA) è invece una estensione del modello vettoriale, che ne riesce a migliorare efficienza ed efficacia. L'efficacia migliora perchè il modello cerca di estrarre informazioni sui legami tra le parole, mentre l'efficienza migliora perchè il reperimento viene effettuato su una matrice di dimensioni minori rispetto alla corrispondente del modello vettoriale: questo è possibile grazie alla decomposizione SVD (*Single Value Decomposition*).

L'importanza del modello LSA è anche dovuta al fatto che negli ultimi anni è stato usato nel reperimento dell'informazione su reti *peer-to-peer* (P2P). In questo contesto i documenti sono distribuiti su vari calcolatori, e quindi l'efficienza diventa molto importante. Così un modello come LSA, che riesce a ridurre le dimensioni della matrice termini-documenti, si presta bene ad essere utilizzato sulle reti P2P, come base per lo sviluppo di modelli specifici. Un esempio di applicazione di questo tipo può essere trovata in [6].

I modelli presentati sono spiegati più in dettaglio in [1].

Capitolo 3

Introduzione

In questo capitolo verranno richiamati i concetti teorici necessari per poter comprendere come il modello funziona, cioè la DFT e qualche cenno di filtraggio dei segnali e di data mining.

3.1 DFT

Per i segnali a tempo continuo è stata introdotta la trasformata di Fourier, ma nel caso di segnali a tempo continuo periodici la trasformata di Fourier si può esprimere in un modo diverso, ricorrendo allo sviluppo in serie di Fourier di un segnale periodico. Nel caso discreto esiste qualcosa di simile, e la trasformata legata ai segnali discreti periodici si chiama DFT (*Discrete Fourier Transform*). Questa trasformata è molto importante, dal momento che è l'unica trasformata calcolabile per via numerica, in modo anche efficiente attraverso una specifica classe di algoritmi, derivati dal paradigma di programmazione *divide and conquer*, detti FFT (*Fast Fourier Transform*).

Si spiegherà ora come si passa dalla trasformata di Fourier per segnali a tempo discreto o DTFT (*Discrete Time Fourier Transform*) alla DFT, senza però scendere troppo nei dettagli.

Si ricorda che l'interesse per la DFT deriva dal fatto che i segnali con cui si rappresentano i termini dell'interrogazione sono essenzialmente segnali a tempo discreto periodici, ma questo sarà chiarito meglio in seguito.

3.1.1 DTFT

La trasformata di Fourier per un segnale a tempo discreto $x(nT)$, o DTFT, è definita

$$\tilde{X}(f) = T \sum_{n=-\infty}^{\infty} x(nT)\, e^{-j2\pi fnT} \tag{3.1}$$

essa è una funzione periodica, di periodo $F_c = \frac{1}{T}$, però è calcolabile solo matematicamente.

3.1.2 DTFT di segnali a durata finita e DFT

Se il segnale $x(nT)$ ha durata finita, cioè $x(nT) = 0 \; \forall n < 0 \vee n \geq N$, la sua DTFT è

$$\tilde{X}(f) = T \sum_{n=0}^{N-1} x(nT) e^{-j2\pi fnT}. \tag{3.2}$$

Il segnale $x(nT)$ è rappresentabile con un calcolatore, ma la sua trasformata no. Si considera allora la periodicizzazione del segnale $x(nT)$, cioè la sua ripetizione periodica. Il segnale ottenuto è

$$\tilde{x}(nT) = \sum_{k=-\infty}^{\infty} x(nT - kNT) \tag{3.3}$$

che è periodico di periodo NT. Se si prova a calcolare ora la DTFT di tale segnale si ottiene

$$\tilde{X}(KF) = T \sum_{n=0}^{N-1} x(nT) e^{-j2\pi KFnT}, F = \frac{F_c}{N} \tag{3.4}$$

che è la DFT del segnale $x(nT)$. In altre parole, avendo un segnale a tempo discreto a durata finita (figura 3.1a), ad esso corrisponde una DTFT non rappresentabile al calcolatore (figura 3.1b). Ma se tale segnale viene ripetuto, rendendolo quindi periodico (figura 3.1c), la DTFT che ne risulta si chiama DFT ed è corrispondente alla DTFT "campionata" a frequenze multiple di $F = \frac{F_c}{N}$ (figura 3.1d).

3.1.3 Dispersione spettrale nella DFT

Uno dei problemi della DFT di un segnale sinusoidale è che se la frequenza f_0 del segnale non è un multiplo intero del quanto di frequenza F, il risultato è una DFT distorta. Nel caso di un segnale sinusoidale come quello analizzato in questa tesi la DFT corretta sarebbe un impulso alla frequenza f_0 del segnale (es. figura 3.2), mentre si potrebbe ottenere una DFT avente qualche componente non nulla in un intorno di f_0 (es. figura 3.3); questo fenomeno, che generalmente si vorrebbe evitare, è noto con il nome di dispersione spettrale, o *spectral leakage*.[2]

Nell'ambito di questa tesi, la dispersione spettrale viene usata come un vantaggio, in quanto la presenza di queste componenti "parassite" può aiutare se nel documento è presente un termine (x_{ij}) che non è proprio quello

[1] La figura è presa da [3].

[2] Maggiori informazioni possono essere trovate in [4]

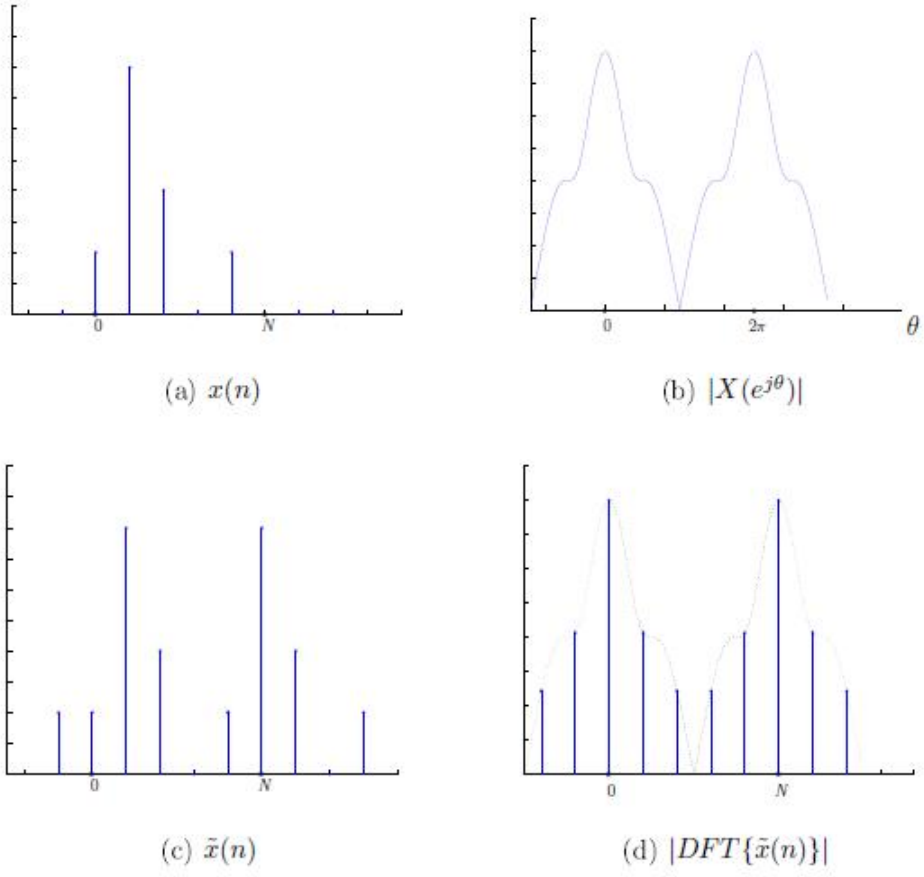


Figura 3.1: Esempio di segnale e trasformate.[1]

presente nella interrogazione (q_t), ma ha una frequenza vicina. Così si avrebbe che l'attenuazione dovuta al termine x_{ij} fa diminuire un po' la potenza associata all'interrogazione, proprio perchè nelle frequenze vicine a quella corrispondente a q_t il modulo della DFT è non nullo. Tale osservazione verrà ripresa durante la spiegazione del funzionamento del modello.

3.1.4 DTFT di segnali a durata infinita

Nel caso di segnali a durata infinita, ci si porta nel caso di segnali a durata finita analizzato in precedenza usando una "finestra". Una finestra in prima approssimazione si può pensare come un segnale che vale 1 in un certo intervallo e 0 altrove (finestra rettangolare, ma ne esistono altre). Così, moltiplicando un generico segnale a tempo discreto per una finestra, si ottiene

[3]La figura è presa da [4].

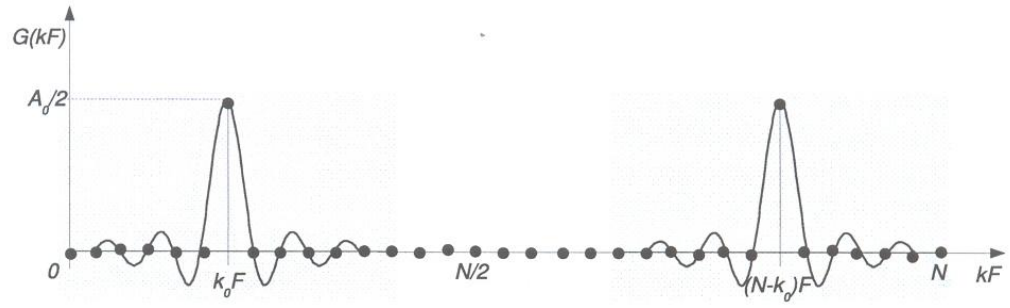


Figura 3.2: DFT di segnale sinusoidale con f_0 multiplo intero di F.[3]

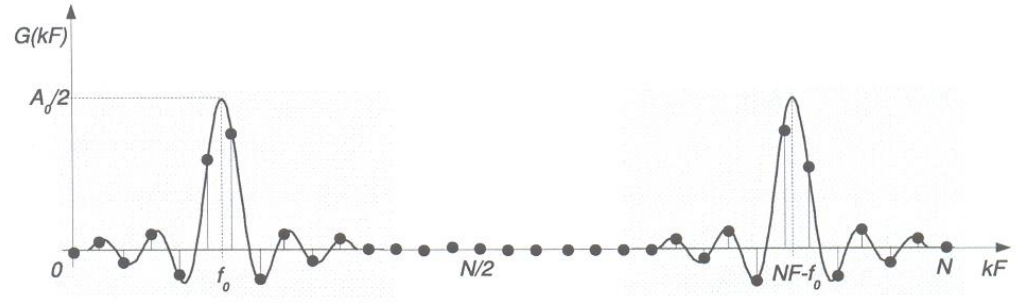


Figura 3.3: DFT di segnale sinusoidale con f_0 multiplo non intero di F.[3]

un segnale uguale a quello originale laddove la finestra vale 1, e 0 altrove. Il risultato è quindi un segnale a durata finita, analizzabile tramite DFT come spiegato precedentemente.

3.1.5 FFT

È possibile calcolare la DFT usando particolari algoritmi veloci detti FFT, in modo da eseguire i calcoli con complessità asintotica $T(N) \in \Theta(N \log N)$, cioè in modo molto efficiente rispetto alla complessità $\Theta\left(N^2\right)$ che si otterrebbe usando la formula così come è. Non verranno trattati comunque i dettagli di questo approccio, ideato da Cooley e Tuckey nel 1965.[4]

Ulteriori approfondimenti sull'argomento possono trovarsi in [2, 3].

3.1.6 Proprietà della DFT e teorema del campionamento

La DFT ha molte proprietà interessanti, che possono essere approfondite in [3, 4], ma per i nostri scopi basta ricordarne tre:

- **Linearità:** $DFT(a \cdot x(nT) + b \cdot y(nT)) = a \cdot \tilde{X}(KF) + b \cdot \tilde{Y}(KF)$, $a, b \in \mathbb{C}$.

[4] In realtà il primo a parlare di tale algoritmo è stato Gauss, nel 1805 circa.

- **Periodicità:** se il segnale $x\,(nT)$ è reale e periodico di periodo N, allora $\tilde{X}\,(-KF) = \tilde{X}\,((N-K)\,F)$.
- **Hermitianità:** se il segnale $x\,(nT)$ è reale, allora $\tilde{X}\,(KF) = \tilde{X}^*\,(-KF)$.

Dalle ultime due si può ricavare che se il segnale $x\,(nT)$ è reale, allora $\tilde{X}\,(KF) = \tilde{X}^*\,((N-K)\,F)$.

Un importante teorema, che servirà in questo lavoro per porre delle condizioni sulla scelta dei alcuni parametri, è il teorema del campionamento o di Whittaker-Nyquist-Kotelnikov-Shannon.[5] Semplificando l'enunciato e adattandolo agli scopi di questa tesi, tale teorema afferma che per poter risalire da un segnale campionato all'originale senza distorsione, la frequenza della componente armonica a frequenza più alta deve essere minore o uguale a metà della frequenza di campionamento. Nel nostro caso, visto che la DFT è in pratica un campionamento della trasformata di Fourier a tempo discreto, e che i segnali analizzati sono segnali sinusoidali, detta F_c la frequenza di campionamento che appare nelle formule per il calcolo della DFT, e detta f_k la frequenza massima tra quelle dei vari segnali sinusoidali sui quali viene calcolata la DFT, deve valere

$$f_k \leq \frac{F_c}{2}. \tag{3.5}$$

3.2 Filtraggio dei segnali

Uno degli argomenti più importanti per quanto riguarda l'elaborazione numerica dei segnali è senz'altro la progettazione di filtri. Un filtro digitale non è altro che un elemento il cui compito è trasformare un segnale di input $x(nT)$ in un altro segnale di output $y(nT)$, dove entrambi i segnali sono a tempo discreto.

La relazione che lega l'ingresso $x(nT)$ con l'uscita $y(nT)$ può essere descritta mediante equazioni alle differenze, e da questa relazione può essere ricavata una caratterizzazione del filtro, detta risposta impulsiva, e indicata solitamente con $h(nT)$.

In pratica, nel caso di sistemi dove l'ingresso e l'uscita sono nulli prima dell'istante 0, una volta ricavata la risposta impulsiva l'uscita è calcolata come la *convoluzione* tra l'ingresso e la risposta impulsiva stessa; più formalmente si ha che

$$y(nT) = x(nT) * h(nT) = \sum_{k=0}^{n} x\,(kT)\,h\,((n-k)\,T)\,u\,(nT) \tag{3.6}$$

dove $u\,(nT)$ rappresenta il gradino unitario, cioè il segnale che vale 0 per $n < 0$ e 1 per $n \geq 1$, e il simbolo $*$ rappresenta l'operatore di convoluzione.

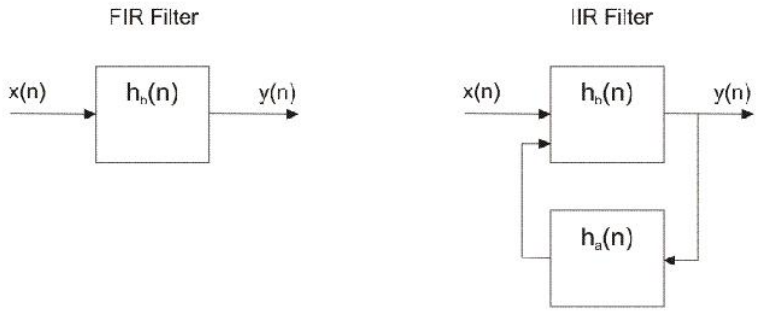
[5] Per ulteriori dettagli consultare [5].

Passando alle trasformate, la relazione ingresso/uscita descritta dall'operazione di convoluzione diventa una moltiplicazione. In altri termini, supponendo che $\tilde{X}(f)$, $\tilde{Y}(f)$ e $\tilde{H}(f)$ siano rispettivamente le trasformate di Fourier a tempo discreto (DTFT) dell'ingresso, dell'uscita e della risposta impulsiva (tale trasformata è detta anche risposta in frequenza), si ha la relazione

$$\tilde{Y}(f) = \tilde{H}(f) \cdot \tilde{X}(f) \tag{3.7}$$

Questo fatto verrà sfruttato quando si dovrà filtrare lo spettro associato all'interrogazione; infatti, senza ricorrere alla convoluzione, ma ragionando nel dominio della frequenza usando la DFT come approssimazione della DTFT, l'operazione sarà molto più immediata.

Una distinzione importante da fare, per quanto riguarda i filtri, è la differenza tra i filtri FIR (*Finite Impulse Response*) e IIR (*Infinite Impulse Response*). La differenza principale è dovuta al fatto che nei primi l'uscita ad un certo istante dipende solo dai valori di ingresso fino a quell'istante, mentre nei secondi l'uscita dipende anche dai valori precedenti dell'uscita. Questo si traduce anche nel fatto che nei filtri FIR la risposta impulsiva ha durata finita, mentre nei filtri IIR ha durata infinita. Inoltre la struttura di un filtro IIR è sempre ricorsiva, mentre quello di un FIR no (a parte alcuni casi particolari), come evidenziato dalla figura 3.4.



(a) (b)

Figura 3.4: Schema a blocchi di filtro FIR (a) e IIR (b).[6]

In questa tesi verrà utilizzato un filtro concettualmente molto vicino a un particolare filtro IIR del secondo ordine chiamato *notch*, o filtro elimina banda, in quanto la sua caratteristica è quella di eliminare lo spettro attorno a una certa frequenza f_0, lasciando quasi inalterato il resto, come mostrato dalla figura 3.5

[6]La figura è reperibile all'URL `http://www.electroportal.net/image.php?id=948`

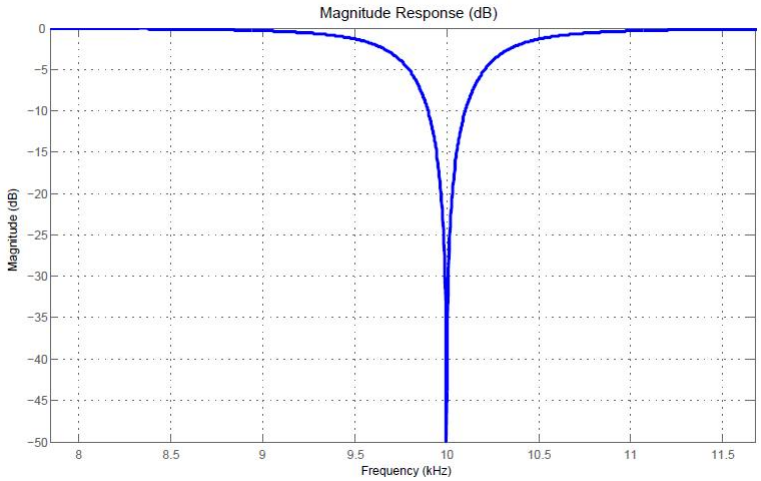


Figura 3.5: Modulo di $\tilde{H}(f)$ per un filtro notch con $f_0 = 10$ kHz.

3.3 Data mining

Il data mining si occupa dell'estrazione di dati interessanti o pattern nascosti in grandi database. Per gli scopi di questa tesi verranno utilizzati solo alcuni concetti, utili per capire come il modello riesca ad associare a ogni termine una frequenza, e con quale criterio si stabiliscono i gruppi di termini con frequenze vicine, che porteranno poi ad un clustering della collezione.

I concetti verranno ora introdotti usando la terminologia generale del data mining, ma si spiegherà come adattarli per poterli usare in questo lavoro; in particolare, definiamo

- l'insieme $I = \{i_1, i_2, \ldots, i_n\}$ degli *item* distinti; gli item rappresentano i termini distinti della collezione;
- la transazione $T \subseteq I$; rappresenta un documento della collezione;
- il dataset D, che è un insieme di transazioni; rappresenta la collezione.

Siano inoltre $A \subseteq I$ e $B \subseteq I$, con $A \cap B = \emptyset$. Si definisce *regola associativa* una scrittura del tipo $A \rightarrow B$. Il significato di questa regola è che nelle transazioni del dataset se è presente l'insieme di item (o itemset) A, con una certa probabilità è presente nella stessa transazione l'itemset B. Nel modello *LSPR* si considerano regole associative che riguardano solo itemset di cardinalità 1, cioè A e B sono termini singoli; allora il significato della regola $A \rightarrow B$ è che nella collezione i documenti contenenti il termine A contengono con una certa probabilità anche il termine B.

Resta però da definire quanto una regola associativa è valida nel dataset. Bisogna guardare infatti in quante transazioni se c'è A allora c'è anche B, e inoltre guardare in quante transazioni del dataset sono presenti A e

B insieme. In altre parole, si deve rispondere a due domande: quale è la probabilità che se in una transazione si osserva l'itemset A allora si osserva anche B? E nella collezione, quale è la probabilità di avere una transazione con A e B insieme? Alla prima domanda si risponde introducendo il concetto di confidenza, alla seconda con il concetto di supporto. Più formalmente, indicando con $|T(X,Y)|$ il numero di transazioni che contengono gli itemset X e Y, e con $|D|$ l'insieme delle transazioni della collezione, definiamo rispettivamente supporto e confidenza della regola associativa $(A \to B)$ le quantità

- $\text{Supp}(A \to B) = P(A \wedge B) = \frac{|T(A,B)|}{|D|}$.
- $\text{Conf}(A \to B) = P(B|A) = \frac{|T(A,B)|}{|T(A)|}$.

In pratica il supporto dice quanto spesso una regola è presente nel dataset, mentre la confidenza indica quanto *forte* è tale regola.

Come già accennato, nel modello *LSPR* le regole associative serviranno a stabilire quanto un termine è legato ad un altro, in modo da poter assegnare frequenze vicine ai termini più legati tra di loro: questo si otterrà proprio usando i concetti di supporto e confidenza, come spiegato nella sezione 4.2.2.2. In particolare, dato $0 \leq \alpha \leq 1$, $\alpha \in \mathbb{R}$ definiamo *attendibilità* di una regola associativa la seguente espressione:

$$\text{Att}(\text{A} \to \text{B}) = \alpha \cdot \text{Supp}(\text{A} \to \text{B}) + (1 - \alpha) \cdot \text{Conf}(\text{A} \to \text{B}). \tag{3.8}$$

Questa misura, come si vedrà in seguito, servirà a definire la qualità delle regole associative, in modo da poter escludere quelle che non raggiungono una certa soglia di attendibilità.

Questo è ciò che occorre sapere per ora riguardo al data mining; per una trattazione più esauriente si rimanda a [8].

Capitolo 4

Metodologia

Durante lo svolgimento di questo lavoro, è stato deciso di mettere a confronto due diversi modelli: il primo si basa esclusivamente sull'utilizzo delle regole associative, mentre il secondo è quello basato su DFT (*LSPR*). In questo capitolo verrà spiegato come funzionano entrambi i modelli, analizzando dettagliatamente ogni componente.

Si ricorda che le prove sono state effettuate sulla collezione CACM: un file contiene tutti i documenti della collezione, un altro contiene tutte le interrogazioni, mentre un ulteriore file contiene i giudizi di rilevanza, indispensabili per quantificare le prestazioni del modello. Questi aspetti verranno chiariti nel capitolo 6.

Una cosa importante da sottolineare è che nell'implementazione si è guardato più all'efficacia che all'efficienza del codice: non si è infatti speso molto tempo nell'ottimizzare il codice per migliorarne la velocità e l'occupazione di memoria, in quanto lo scopo di questo lavoro è la valutazione della qualità del reperimento per il modello *LSPR*, con la convinzione che solo a seguito di una buona efficacia ha senso garantire anche l'efficienza.

4.1 Indicizzazione e rappresentazione del contenuto informativo

Molte delle parti sviluppate per il modello basato su regole associative sono utilizzate anche da *LSPR*, così inizialmente verranno descritte le parti comuni, e poi verranno spiegate separatamente le parti relative al modello specifico.

4.1.1 Rimozione delle stop word dalla collezione

La prima fase riguarda la rimozione delle *stop word*, cioè le parole che non forniscono un contributo informativo rilevante (per esempio gli articoli). Per

la rimozione di tali parole, si è utilizzata una lista di termini in inglese che deriva dalla fusione di alcune liste, in particolare quella del software Terrier,[1] quella presente presso il sito del complesso biologico A. Vallisneri di Padova[2] e quella creata da Gerard Salton e Chris Buckley alla Cornell University nell'ambito del sistema sperimentale di information retrieval SMART.[3]

4.1.2 Stemming

La seconda operazione effettuata è la riduzione delle parole alle loro radici, o *stemming*. Questa operazione consiste nell'estrarre, da ogni parola rimasta nella collezione dopo la fase precedente, una sottostringa chiamata *stem*, che ne rappresenta la radice linguistica. Questo viene fatto perchè spesso parole che sono rappresentate dallo stesso stem hanno un qualche legame, come ad esempio per *computer* e *computing*, il cui stem può essere *comput*.

Al momento dell'interrogazione da parte dell'utente è utile poter reperire anche quei documenti che non hanno il termine cercato, ma altri termini con lo stesso stem. Questo approccio è inoltre utile in fase di indicizzazione, perchè non ci sarà un entry per ogni termine della collezione, ma tutti i termini aventi lo stesso stem saranno rappresentati proprio dallo stem: si avrà così una riduzione delle dimensioni dell'indice.

Per poter calcolare gli stem dei vari termini in questo lavoro si è usato l'algoritmo di Porter, ideato nel 1980, che è uno dei più famosi per la lingua inglese.[4]

4.1.3 Calcolo della matrice termini-documenti e pesatura

In questa fase è stata creata una matrice, in cui le righe corrispondono ai termini della collezione, mentre le colonne ai documenti. In particolare, chiamando questa matrice T, si ha che $T[i, j]$ rappresenta il peso del termine i nel documento j.

Il peso è calcolato con lo schema TF-IDF normalizzato, che deriva dalla combinazione degli schemi TF e IDF. Per comprenderne il funzionamento, supponiamo di voler calcolare il peso del termine i nel documento j (indicato in seguito con w_{ij}). TF (*Term Frequency*) rappresenta il numero di occorrenze di i in j, e indichiamo questo valore con TF_{ij}. IDF (*Inverse Document Frequency*) viene usato per considerare il fatto che se in una interrogazione si ha un termine che appare in quasi tutti i documenti, esso dovrebbe avere una importanza minore rispetto a un termine che appare in meno documenti. Detti $|C|$ il numero di documenti della collezione e n_i il

[1] Software reperibile all'URL `http://ir.dcs.gla.ac.uk/terrier/`

[2] Reperibile all'URL `http://www-fog.bio.unipd.it/waishelp/stoplist.html`

[3] Reperibile all'URL `http://www.lextek.com/manuals/onix/stopwords2.html`

[4] Si è usata una implementazione di questo algoritmo in linguaggio Java, reperibile all' URL `http://snowball.tartarus.org/dist/libstemmer_java.tgz`

numero di documenti contenenti il termine i, IDF_{ij} è definito come

$$\mathrm{IDF}_{ij} = \begin{cases} \log(|C|/n_i) & n_i > 0 \\ 0 & n_i = 0. \end{cases} \tag{4.1}$$

Lo schema TF-IDF considera entrambi i contributi, e quindi si ha che $\mathrm{w}_{ij} = \mathrm{TF}_{ij} \cdot \mathrm{IDF}_{ij}$. In realtà questo valore viene normalizzato, così il peso effettivamente usato è

$$\hat{\mathrm{w}}_{ij} = \frac{\mathrm{w}_{ij}}{\sqrt{\sum_i \mathrm{w}_{ij}^2}}. \tag{4.2}$$

Un ulteriore motivo dell'utilizzo di IDF nella pesatura è dovuto al fatto che questo bilancia il contributo dato dal supporto, che è tanto maggiore quanti più documenti contengono il termine.

4.1.4 Analisi delle query

A questo punto si passa all'analisi delle interrogazioni: vengono rimosse le stop word, si applica lo stemming di Porter e si ordinano alfabeticamente i termini delle query rimuovendo i duplicati, ed effettuando una pesatura IDF (solo per *LSPR*), usando la formula dall'equazione (4.1). D'ora in poi con Q si intende la query risultante dopo queste operazioni.

4.1.5 Calcolo delle regole associative

L'ulteriore passo da effettuare è il calcolo di supporto e confidenza delle regole associative, come definiti nella sezione 3.3. Le regole associative valutate sono quelle del tipo $\mathrm{A} \rightarrow \mathrm{B}_j$, $\mathrm{A} \neq \mathrm{B}_j$, con A termine dell'interrogazione presenta nella collezione e B_j termine della collezione. Avendo a disposizione la matrice termini-documenti creata nella fase precedente, il calcolo risulta abbastanza semplice; inoltre una volta calcolati supporto e confidenza, si calcola l'attendibilità delle regole associative, e solo quelle con un livello di attendibilità superiore a una certa soglia vengono considerate. Nel codice sviluppato tale soglia è stata impostata a 0.03, mentre i pesi di supporto e confidenza per il calcolo dell'attendibilità sono entrambi pari 0.5 (il valore α nell'equazione (3.8) è 0.5).

4.1.6 Calcolo dei gruppi associativi

Una volta calcolate le regole associative, e rimosse quelle che non presentano una attendibilità sufficiente, vengono creati i *gruppi associativi*, cioè gruppi costituiti da un termine principale (il termine A della regola associative) e da un insieme di termini secondari (i termini B_j della regola associativa). Ogni gruppo rappresenta così un insieme di termini correlati tra di loro, ognuno con grado diverso definito dall'attendibilità: in sostanza ciò che si è fatto è un clustering dei termini della collezione.

4.1.7 Calcolo delle collezioni ridotte

Il passo ulteriore è quello di calcolare, per ogni interrogazione, un sottoinsieme dell'intera collezione su cui effettuare il reperimento: questa si chiama *collezione ridotta*.

Ciò che si fa in pratica è, per ogni termine q_i dell'interrogazione presente anche nella collezione, considerare il gruppo associativo avente come termine principale q_i, e aggiungere alla collezione ridotta tutti i documenti che contengono almeno uno dei termini del gruppo. Questo procedimento è spiegato meglio dallo pseudocodice seguente:

Algorithm: Calcolo_collezione_ridotta
Input: query Q, *gruppi associativi di* Q
Output: collezione ridotta *coll*

```
1  coll ← ∅
2  for each termine q_i ∈ Q
3        do
4           if esiste un gruppo associativo G con termine principale q_i
5                 do coll ← coll ∪ {documenti con almeno un termine di G}
6           end if
7  end for
8  return coll
```

Fino a questo punto le operazioni descritte sono effettuate sia dal modello di riferimento basato solo sulle regole associative, sia dal modello *LSPR*. Si presentano ora le modalità con cui il modello di riferimento calcola il ranking dei documenti, e successivamente si esporrà come invece procede il modello *LSPR*.

4.2 Modelli di reperimento

Si presenteranno ora le caratteristiche principali dei due modelli di reperimento messi a confronto: il primo si basa sulle regole associative, il secondo è *LSPR*, ed è basato sulla DFT.

4.2.1 Modello basato su regole associative

Nel modello di riferimento, basato solamente sulle regole associative, si hanno ora tutti gli elementi per poter calcolare il ranking dei documenti. Infatti, ad ogni interrogazione i documenti restituiti saranno quelli della collezione ridotta associata; resta solo da stabilire con che ordine restituirli.

L'idea è quella di attribuire a ogni documento un punteggio (*score*), in modo che più un documento ha score elevato, prima appare nell'elenco dei risultati. Per l'attribuzione dello score, si procede nel modo seguente:

per ogni termine dell'interrogazione (chiamiamolo per semplicità A), se nel documento è presente proprio quel termine (o meglio lo stem), allora lo score viene incrementato del peso di A; se non è presente il termine esatto ma sono presenti termini che erano la parte destra di una regola associativa (i vari B_j di $A \rightarrow B_j$), allora si incrementa lo score del peso dei vari termini B_j, moltiplicati ognuno per l'attendibilità della relativa regola associativa. Lo pseudocodice seguente chiarisce meglio il funzionamento dell'algoritmo relativamente a una interrogazione Q.

Algoritm: Ranking_no_DFT
Input: query Q, *gruppi associativi di Q*
Output: array *result* contenente i documenti ordinati per importanza decrescente

```
1  coll ← Calcolo_collezione_ridotta(Q, gruppi associativi di Q)
2  for each documento X_i ∈ coll
3        do
4           score[i] ← 0
5           for each A ∈ Q
6                 do
7                    ⋆ sia G il gruppo associativo con termine principale A ⋆
8                    if A ∈ X_i \\ ho il termine nel documento
9                      then
10                          score[i] ← score[i]+ peso di A in X_i
11                     else
12                          score[i] ← score[i] + Σ_{j|B_j∈G\A} peso di B_j · Att(A → B_j)
13                   end if
14          end for
15 end for
16 ⋆ Ordina i documenti secondo score decrescente e salvali in result ⋆
17 return result
```

4.2.2 Modello basato su DFT: *LSPR*

Di seguito verrà esposto come il modello *LSPR* effettua il ranking dei documenti: questa è la parte più importante del modello, che lo contraddistingue da quello basato solo sulle regole associative. La trattazione sarà divisa in 3 parti, dove verranno spiegati rispettivamente la trasformazione da interrogazione a spettro, la trasformazione da documento a filtro e il ranking finale.

4.2.2.1 Trasformazione interrogazione spettro

Si supponga di avere una interrogazione Q, e la collezione ridotta dei documenti relativi a tale interrogazione, come è stata definita precedentemente.

Il primo passo è quello di stabilire i vari parametri necessari per il calcolo della DFT associata all'interrogazione. In particolare, si devono definire il numero di campioni N della DFT e il quanto di frequenza F (così rimangono definiti anche i valori F_c, T e T_p, date le relazioni presentate nella notazione), e anche le frequenze e le ampiezze che servono a costruire i segnali sinusoidali associati a ogni termine dell'interrogazione.

Per quanto riguarda il valore di F, esso viene definito pari a 2 Hz, e questo valore rimane fisso. Il motivo di tale scelta è che si vuole garantire l'effetto di dispersione spettrale. Come spiegato nel capitolo 3, tale fenomeno si verifica quando la frequenza f_0 del segnale sinusoidale non è un multiplo intero del quanto di frequenza F: così, se F vale 2 Hz, basta che le varie frequenze dei termini dell'interrogazione siano numeri dispari, per soddisfare questa condizione.

Per spiegare come viene calcolato N, bisogna prima spiegare la struttura dello spettro associato all'interrogazione e fornire alcune precisazioni sul calcolo della DFT. Per ogni termine dell'interrogazione, si è scelto di destinare 300 punti dello spettro, e di far cadere il picco relativo al termine intorno al punto 200;[5] il motivo di questa scelta verrà chiarito tra breve. Il numero di punti per una query Q sarebbe pari a $300|Q|$, e data la relazione $F_c = N \cdot F$, si avrebbe una frequenza di campionamento di $300|Q|F$. La frequenza massima tra i vari segnali sinusoidali è però pari a $(300\,(|Q| - 1) + 200)\,F$, violando così il teorema del campionamento, che impone che la frequenza massima sia minore o uguale a metà della frequenza di campionamento. Per ovviare a questo problema, basterebbe porre $F_c = 2 \cdot 300|Q|F$ (cioè $N = 2 \cdot 300|Q|$), però si vuole tenere un altro intervallo di protezione da 300 punti dopo l'ultimo (non assegnato ad alcun termine) in modo da garantire che le componenti della DFT dell'ultimo termine dell'interrogazione siano praticamente nulle prima della metà dello spettro; inoltre per il calcolo efficiente della DFT, N deve essere potenza di 2. La formula finale per il calcolo di N è quindi

$$N = 2^{\lceil \log_2 (2 \cdot 300(|Q|+1)) \rceil}. \tag{4.3}$$

Per comprendere meglio il significato di questa espressione, supponiamo di avere una interrogazione con 2 termini aventi stesso peso. La formula (4.3) fornirebbe un valore di N pari a $2^{\lceil \log_2 (2 \cdot 300(2+1)) \rceil} = 2^{\lceil \log_2 (1800) \rceil} = 2^{11} = 2048$. La figura 4.1 mostra in maniera più chiara come sono suddivisi i punti dello spettro associato all'interrogazione.

Resta da chiarire perchè si è scelto di porre il picco relativo ai termini dell'interrogazione intorno al punto 200 di ogni gruppo; il motivo di questa scelta è che, come dimostrato da varie prove effettuate con MATLAB®, allontanandosi di circa 100 punti sia a destra che a sinistra rispetto al punto dove è presente il picco, il modulo dello spettro della DFT vale meno dell'1%

[5] **Osservazione importante:** come spiegato nella Notazione, con Q si intende la query tolti i termini che non sono presenti nella collezione.

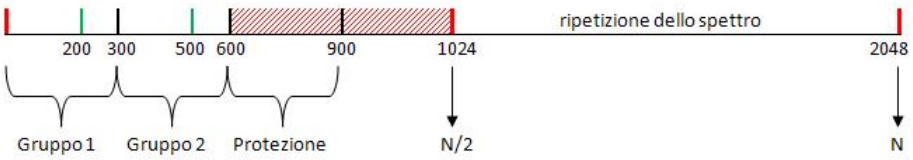


Figura 4.1: Suddivisione dei punti dello spettro dell'interrogazione

del valore di picco. Così intorno ai 100 e 300 punti di ogni gruppo il modulo della DFT può considerarsi quasi esaurito; lo spazio tra 0 e 100 punti serve per contenere sia i valori della parte sinistra della DFT del gruppo in esame (anche se molto attenuati), sia eventuali componenti della parte destra della DFT del gruppo precedente (quelli presenti dopo i 300 punti). Per comprendere meglio questo discorso, si osservi la figura 4.2, che rappresenta lo spettro associato all'interrogazione dell'esempio precedente, come verrebbe effettivamente calcolato dall'algoritmo sviluppato (in ascissa ci sono gli indici dei punti, e non le frequenze; per le frequenze basta moltiplicare l'indice del punto per F, in questo caso per 2 Hz). Si nota che lo spettro è simmetrico rispetto a $N/2$, come ci si aspettava, e che sono presenti dei picchi intorno ai 200 punti (primo gruppo) e ai $300 + 200 = 500$ punti (secondo gruppo).

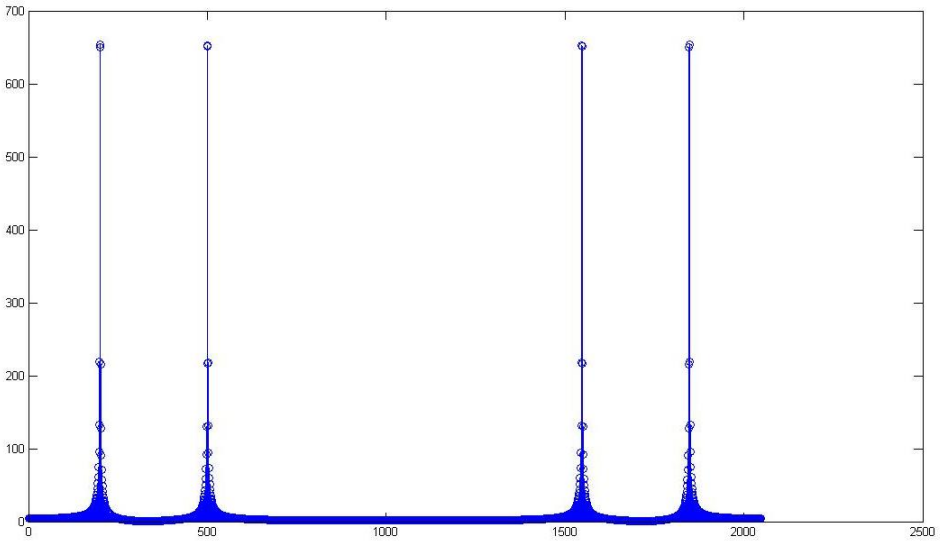


Figura 4.2: Spettro per una query con 2 termini

Si nota che il picco massimo vale circa 650; intorno ai punti 100 e 300, il

modulo della DFT dovrebbe valere meno di 6.5, e infatti questo è dimostrato dalle figure 4.3a e 4.3b. In figura 4.4 è invece possibile avere una visione più chiara dell'andamento dello spettro in un gruppo.

L'ultima cosa che rimane da spiegare sulla costruzione dello spettro riguarda la definizione dei segnali sinusoidali necessari per ottenere i campioni sui quali calcolare la DFT. Ad ogni termine q_i dell'interrogazione, viene associato un segnale sinusoidale avente forma

$$A_i \sin\left(2\pi f_i n T\right). \tag{4.4}$$

Si è deciso di porre A_i pari al peso IDF del termine i della query nella collezione, anche se si poteva pensare di usare altre pesature, come la TF o la binaria; l'efficacia migliore si è però ottenuta con la pesatura IDF. Per quanto riguarda la frequenza, essa deve corrispondere circa al punto 200 di ogni gruppo, come spiegato precedentemente; però bisogna ricordare che per garantire l'effetto di dispersione spettrale tale frequenza deve essere dispari. La formula che permette di calcolare la frequenza f_i dell'i-esimo termine dell'interrogazione è allora data da

$$f_i = \left(300 \cdot (i-1) + 200\right) F + 1, \quad i = 1, 2, \ldots, |Q|. \tag{4.5}$$

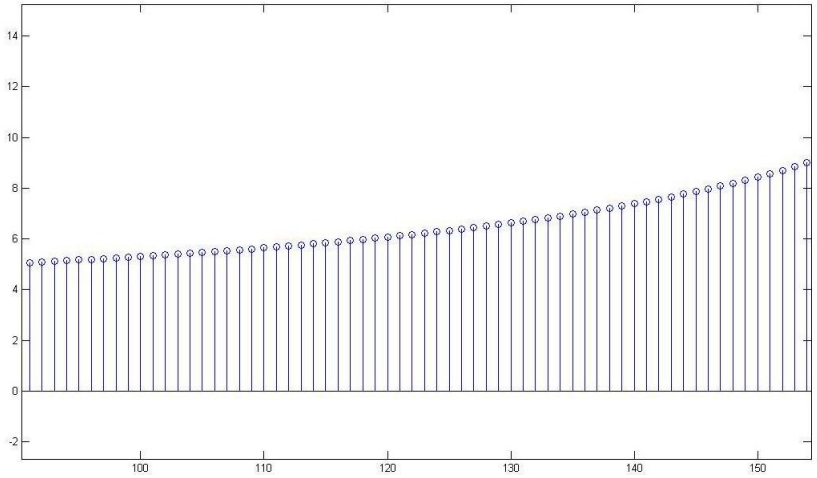
Nell'esempio precedente, ricordando che F vale 2 Hz, la frequenza del primo termine sarebbe stata di 401 Hz, mentre quella del secondo di 1001 Hz. Una volta ottenute le frequenze di tutti i termini, basta valutare la somma dei $|Q|$ segnali sinusoidali ottenuti su N punti, per avere il vettore su cui calcolare la DFT. Chiamando questo vettore *campioni*, si ha che

$$\mathit{campioni}[n] = \sum_{i=1}^{|Q|} A_i \sin\left(2\pi f_i n T\right), \quad n = 1, 2 \ldots, N. \tag{4.6}$$

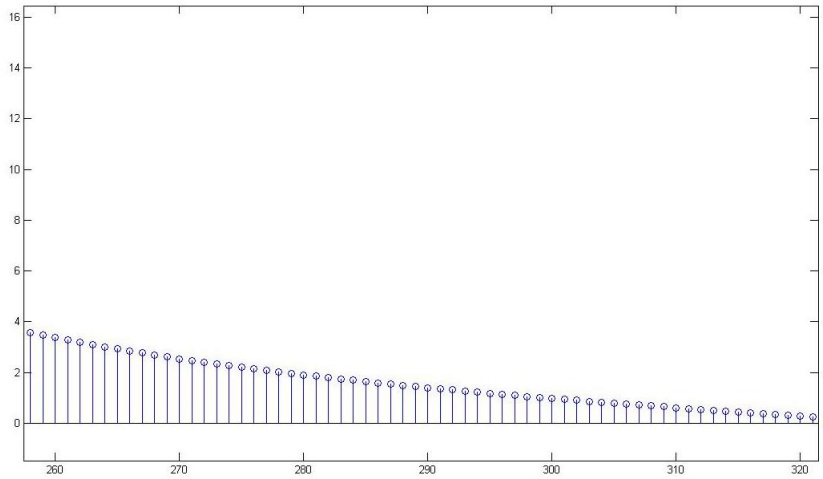
Ricordando che $F = 2$ Hz e che $T = \frac{1}{NF}$, possiamo semplificare la formula precedente in

$$\mathit{campioni}[n] = \sum_{i=1}^{|Q|} A_i \sin\left(\frac{\pi f_i n}{N}\right), \quad n = 1, 2 \ldots, N. \tag{4.7}$$

L'unica cosa che manca è l'algoritmo per il calcolo della DFT; non viene qui riportato lo pseudocodice, in quanto è un algoritmo conosciuto in letteratura, ma alcuni dettagli implementativi verranno riportati nel capitolo 5. Per ora basta sapere che tale algoritmo, dato un vettore di campioni, restituisce la DFT relativa a metà spettro (data la simmetria non serve valutare tutto lo spettro). A questo punto si possono riassumere le operazioni che portano alla creazione del modulo dello spettro associato alla query con lo pseudocodice seguente, dove DFT(x) rappresenta l'algoritmo che fornisce la DFT del vettore x in ingresso.



(a)



(b)

Figura 4.3: Modulo della DFT intorno ai punti 100 (a) e 300 (b).

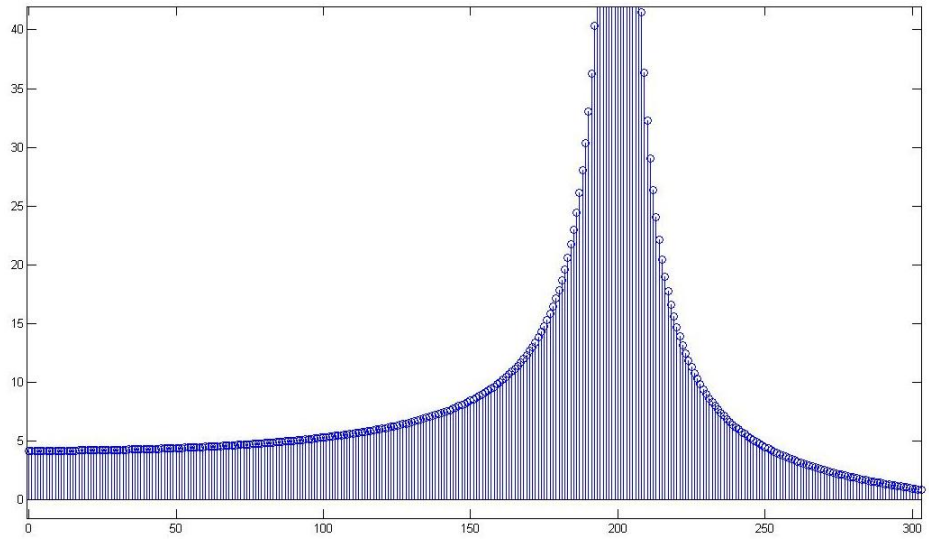


Figura 4.4: Dettaglio dello spettro dell'interrogazione in corrispondenza di un gruppo

Algoritm: Calcola_spettro
Input: query Q (con A_i peso del termine q_i), quanto di frequenza F
Output: array *spettro* contenente i valori del modulo dello spettro della query

1 **for each** termine $q_i \in Q$
2 **do**
3 $\mathrm{f}_i \leftarrow (300 \cdot (i-1) + 200)\, F + 1$
4 **end for**
5 **for** $n \leftarrow 1$ **to** N
6 **do**
7 $$campioni[n] = \sum_{i=1}^{|Q|} A_i \sin\left(\frac{\pi f_i n}{N}\right)$$
8 **end for**
9 *spettro* $\leftarrow$|DFT(*campioni*)|
10 **return** *spettro*

4.2.2.2 Trasformazione documento filtro

Ora che si è spiegato come passare da una interrogazione allo spettro relativo, si deve definire come un documento attenua tale spettro, o in altra parole che tipo di filtro rappresenta il documento stesso. Come già spiegato nel capitolo 3, un documento rappresenta un insieme di filtri notch (o multi-notch), e dal capitolo 1 dovrebbe essere chiaro che i termini del documento

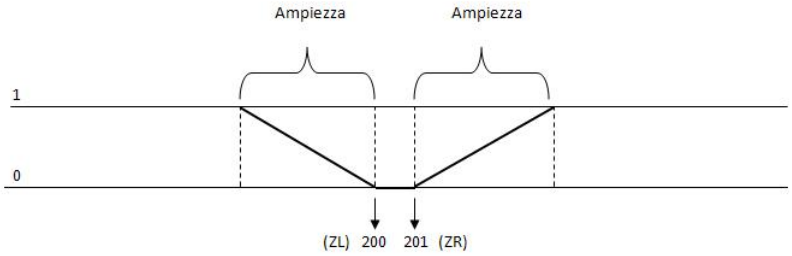


Figura 4.5: Rappresentazione schematica del filtro notch per il primo gruppo (punti 0-300)

e i relativi pesi saranno i responsabili dei parametri di tali filtri. Il filtro che viene utilizzato non è esattamente uguale a quello di figura 3.5, ma ha una forma triangolare, come evidenziato dalla figura 4.5. Qui si vede che il filtro annulla due componenti dello spettro (quelle nei punti 200 e 201 della figura), mentre moltiplica per valori che vanno da 0 a 1 le componenti a destra del punto 200 e a sinistra del punto 201, e in questa operazione sono coinvolti un numero di punti pari all'ampiezza del filtro. La figura rappresenta cosa succede nel primo gruppo (da 0 a 300 punti), ma la situazione si ripete allo stesso modo negli altri intervalli.

Resta da spiegare come determinare i valori di ampiezza e i punti dove il filtro attenua completamente lo spettro. Supponiamo di avere un documento X_j della collezione ridotta e una interrogazione Q, della quale si è calcolato lo spettro; per ogni termine della interrogazione, che corrisponde a un intervallo di 300 punti sullo spettro, il documento può produrre dei filtri. Sia i l'intervallo che si sta analizzando, corrispondente quindi al i-esimo termine dell'interrogazione; si indicherà questo termine con A per semplicità. Se nel documento è presente tale termine, allora viene creato un filtro come quello di figura 4.5, che azzera lo spettro in corrispondenza dei punti $300 \cdot (i-1) + 200$ (ZL o *Zero Left*) e $300 \cdot (i-1) + 201$ (ZR o *Zero Right*), dove sono presenti i picchi massimi: in pratica la frequenza f_0 del filtro, che è la frequenza associata al termine A del documento, è uguale alla frequenza del segnale sinusoidale associato all'intervallo i. Per quanto riguarda l'ampiezza, essa si calcola moltiplicando una quantità fissa chiamata *selettività* (impostata sperimentalmente a 24) per il peso del termine nel documento, e arrotondando questo valore all'intero più vicino.

Se nel documento non c'è il termine A, ma ci sono alcuni termini B_j del gruppo associativo di A, allora vengono creati dei filtri, tanti quanti sono i termini B_j, aventi ampiezza determinata come per il caso precedente (usando il peso di B_j), ma i punti di attenuazione massima ZL e ZR (quindi la frequenza f_0 del filtro) questa volta dipendono dall'attendibilità della regola associativa $A \rightarrow B_j$. Detto G il gruppo associativo con termine principale A,

si calcola la somma delle attendibilità di tutte le regole associative del tipo $A \rightarrow B_j$, dove B_j sono termini del documento che appartengono a G. Viene poi calcolato per ogni termine B_j il rapporto, in percentuale, tra l'attendibilità di $A \rightarrow B_j$ e la somma appena calcolata. Questo valore, moltiplicato per 100, e arrotondato per difetto, ci dice in quale punto dell'intervallo 100-200 di ogni gruppo c'è ZL, e una volta calcolato questo ZR=ZL+1.

Infine, se nel documento non c'è il termine A e nemmeno termini B_j del relativo gruppo associativo, lo spettro in quell'intervallo non viene filtrato.

Verrà ora presentato lo pseudocodice che riassume i passaggi appena descritti, realizzando il filtraggio dello spettro di una interrogazione dato un documento della collezione ridotta. Si suppone di avere a disposizione un algoritmo Filtra(*spettro*, ZL, *intervallo*, *ampiezza*), che dati come parametri lo spettro dell'interrogazione, il punto ZL (ZR è calcolato come ZL+1), l'intervallo di 300 punti dove il filtro deve effettuare il filtraggio e l'ampiezza, realizza la funzione descritta dalla figura 4.5, restituendo lo spettro filtrato, e l'algoritmo Round(x), che arrotonda il numero reale x all'intero più vicino.

Algoritm: Filtraggio
Input: query Q, *gruppi associativi di* Q, documento X_j della collezione ridotta associata
Output: spettro della query filtrato *spettro_filtrato*

```
1  spettro_filtrato ← Calcola_spettro(Q)
2  for i ← 1 to |Q|
3        do
4           ⋆ sia G il gruppo associativo con termine principale q_i ⋆
5           if q_i ∈ X_j \\ ho il termine nel documento
6              then
7                   ZL ← 300 · (i − 1) + 200
8                   ampiezza ← Round(selettività · peso di q_i in X_j)
9                   spettro_filtrato ← Filtra(spettro_filtrato, ZL, ampiezza, i − 1)
10             else
11                  H ← {G ∩ X_j} \q_i
12                  den ← Σ_{j|B_j∈H} Att(q_i → B_j)
13                  for each B_j ∈ H
14                       do
15                          ZL ← ⌊300 · (i − 1) + 100 + 100 · Att(q_i→B_j)/den⌋
16                          ampiezza ← Round(selettività · peso di B_j in X_j)
17                          spettro_filtrato ← Filtra(spettro_filtrato, ZL, ampiezza, i − 1)
18                  end for
19          end if
20 end for
21 return spettro_filtrato
```

Per fare un esempio numerico, supponendo che nel documento non ci sia il termine A ma due termini B_1 e B_2, e che l'attendibilità di $A \rightarrow B_1$ sia 0.4 mentre quella di $A \rightarrow B_2$ sia 0.3, e supponendo che il termine A sia il terzo dell'interrogazione, si ha che il filtro associato a B_1 ha un valore di ZL pari a $\left\lfloor 2 \cdot 300 + 100 + \frac{0.4}{0.4+0.3} \cdot 100 \right\rfloor = 757$, mentre ZR $= 758$. I picchi dello spettro sono invece presenti intorno ai punti 800 e 801.

4.2.2.3 Ranking

Data l'interrogazione Q e la collezione ridotta associata, si esegue l'algoritmo Filtraggio appena descritto e si ottiene, per ogni documento della collezione ridotta, uno spettro filtrato. A questo punto non resta che calcolare la potenza di questi spettri (che non è però la vera potenza, come spiegato nel capitolo 1), e ordinare i documenti secondo potenza crescente dei relativi spettri filtrati. Lo pseudocodice seguente spiega il processo di ranking appena descritto.

Algoritm: Ranking_DFT
Input: query Q, *gruppi associativi di* Q
Output: array *result* contenente i documenti ordinati per importanza decrescente

1 *coll* ← Calcolo_collezione_ridotta(Q, *gruppi associativi di* Q)
2 **for each** documento $X_i \in$ *coll*
3 **do**
4 *spettro_filtrato* ← Filtraggio(Q, *gruppi associativi di* Q, X_i)
5
6 $potenza[i] \leftarrow \sum_{j=1}^{|spettro\_filtrato|} spettro\_filtrato[j]$
7 **end for**
8 ⋆ Ordina i documenti secondo potenza crescente e salvali in *result* ⋆
9 **return** *result*

Capitolo 5

Implementazione

In questo capitolo verrà presentata la realizzazione in Java dei modelli sviluppati, spiegando in particolare le modalità di interazione tra le varie classi sviluppate.

Tutti gli algoritmi il cui pseudocodice è stato presentato nel capitolo 4 sono stati implementati in Java, per poter così confrontare le prestazioni del modello basato su regole associative con quelle del modello *LSPR*.

Gli strumenti utilizzati sono Java SDK versione 1.6 e l'ambiente di sviluppo BlueJ versione 2.2.1.[1]

Il progetto, chiamato *jlspr* (acronimo di *java least spectral power ranking*), consiste in 6 classi:

- `jlspr`: la classe principale del progetto;
- `Index`: la classe che si occupa dell'indicizzazione della collezione;
- `QueryProcessing`: la classe che analizza le interrogazioni;
- `AssociativeRules`: la classe che si occupa delle regole associative e del ranking per il modello basato sulle regole associative;
- `DFT`: la classe con i metodi per il calcolo della DFT e che si occupa del ranking per il modello *LSPR*;
- `TrecEval`: la classe che usando il software Trec Eval (di cui si parlerà nel capitolo 6) calcola le prestazioni dei vari modelli sulla collezione sperimentale CACM.

Oltre a queste classi è presente il pacchetto che fornisce i metodi necessari per l'applicazione dello stemming di Porter, come spiegato nella sezione 4.1.2.

Le varie classi e il pacchetto per lo stemming (directory `org`) sono all'interno di un directory dal nome `jlspr`, il cui contenuto è visualizzato da

[1] Il software è reperibile all'URL `http://www.bluej.org/`

BlueJ come illustrato dalla figura 5.1. Al suo interno sono presenti anche altri directory:

- `collection`: qui va inserito il file `cacm.txt`, contenente la collezione sperimentale CACM;
- `index`: qui vengono salvati alcuni file relativi all'indice creati durante l'esecuzione del programma; in particolare, il file `cacm_stopw.txt` che contiene la collezione dopo la rimozione delle stopword, il file `cacm_stopw_stem.txt` che si ottiene applicando lo stemming di Porter al file `cacm_stopw.txt`, e il file `termini.txt` che contiene l'elenco dei termini distinti della collezione;
- `query`: qui va inserito il file `query.txt` contenente le interrogazioni per la collezione sperimentale CACM. Inoltre durante l'esecuzione del programma vengono creati i file `query_stopw.txt` che contiene le interrogazioni dopo la rimozione delle stopword, il file `query_stopw_stem.txt` che si ottiene applicando lo stemming di Porter al file `query_stopw.txt`, il file `query_stop_stem_order_nodupl.txt` che si ottiene dal precedente ordinando alfabeticamente i termini delle interrogazioni ed eliminando i duplicati, e il file `termini_q.txt` che contiene l'elenco dei termini distinti delle interrogazioni;
- `stopword`: qui va inserito il file `stop-word.txt` contenente la lista delle stop word;
- `trec_eval`: qui va inserito il file `qrels.txt`, contenente i giudizi di rilevanza per la collezione CACM. Inoltre sono presenti i file del software Trec Eval e il file batch `trec_eval.bat` appositamente creato per salvare i risultati. Al momento del salvataggio dei risultati, in questo directory saranno presenti il file `risultati.txt`, contenente l'elenco dei documenti reperiti in risposta alle interrogazioni presenti in `query.txt` (formattato per l'utilizzo con Trec Eval), e il file `prestazioni_ARS.txt` o `prestazioni_DFT.txt`, a seconda che sia stato utilizzato il modello basato su regole associative o *LSPR*, contenente le prestazioni del modello (di questo si parlerà nel capitolo 6);[2]
- `doc`: contiene la documentazione del progetto in formato Javadoc.

Tutto il codice presente risulta commentato; l'unica precisazione da fare riguarda l'algoritmo che calcola la DFT. Questo algoritmo sfrutta il fatto

[2]Prima di eseguire il programma, è bene rimuovere o rinominare i file `risultati.txt`, `prestazioni_ARS.txt` e/o `prestazioni_DFT.txt`. In questo directory sono presenti anche i file `prestazioni_ARS_finali.txt` e `prestazioni_DFT_finali.txt`, contenenti l'output di Trec Eval per i due modelli (per maggiori informazioni su Trec Eval consultare la sezione 6.1.1), e i file `risultati_ARS_finali.txt` e `risultati_DFT_finali.txt` contenenti i documenti reperiti dai modelli nel formato corretto per l'utilizzo con Trec Eval.

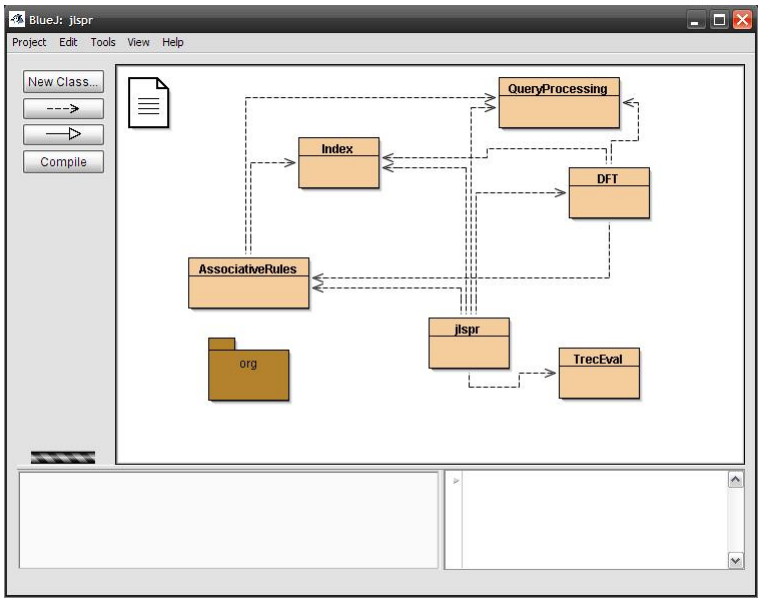
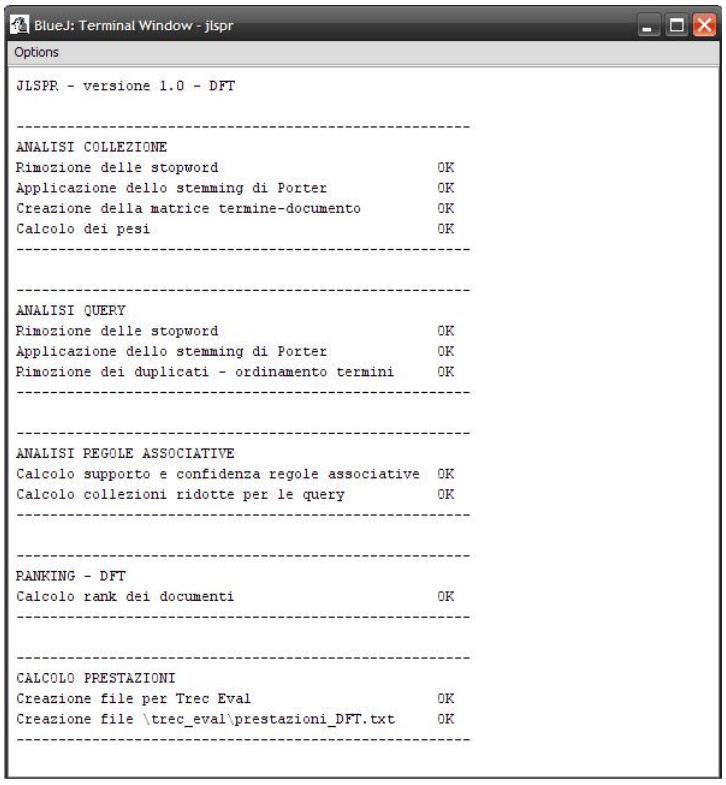


Figura 5.1: Rappresentazione delle varie componenti del progetto con l'ambiente BlueJ.

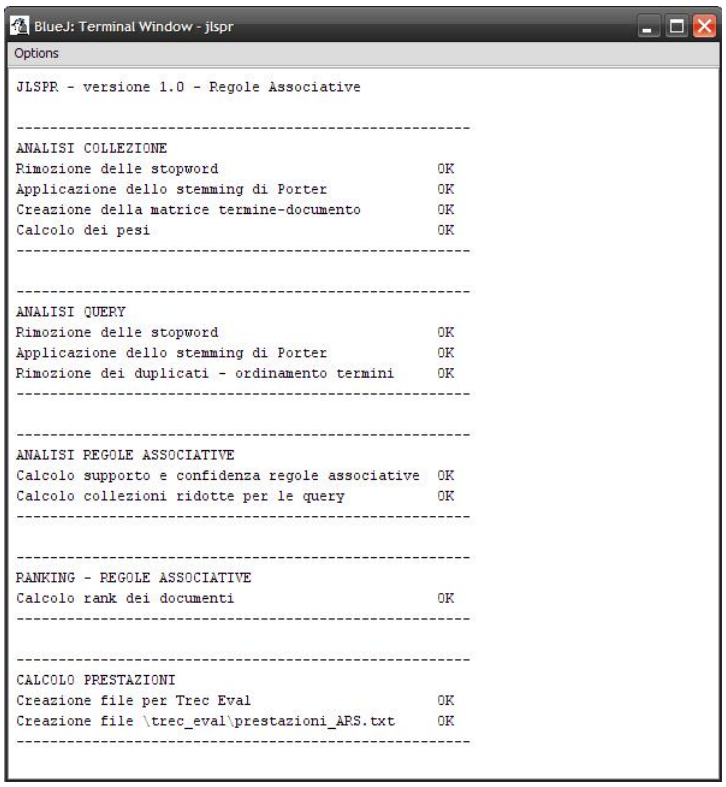
che i campioni su cui calcolare la DFT sono numeri reali, che portano a una simmetria dello spettro rispetto a $N/2$ (come spiegato nella sezione 3.1.6): viene calcolato allora solo metà dello spettro. Inoltre, a differenza dell'algoritmo di Cooley e Tuckey, questo è un algoritmo iterativo e non ricorsivo. Il codice dell'algoritmo presente nella classe DFT è una conversione di quello in linguaggio C che si trova in [7], a parte alcuni piccoli miglioramenti.

Per il corretto funzionamento del software, bisogna aumentare le dimensione dell'heap di Java, in quanto viene creata una matrice termini-documenti direttamente in memoria principale (come si è già detto la velocità e l'occupazione di memoria sono stati due obiettivi secondari). Per ovviare al problema in fase di esecuzione basta fornire la stringa "`-Xms512M -Xmx512M`" come parametro al comando `java`, in modo da impostare le dimensioni dell'heap a 512 MB. Se si usa BlueJ in ambiente windows, bisogna aggiungere tale stringa agli entry `bluej.windows.vm.args` e `bluej.vm.args` del file `bluej.defs` contenuto nel directory `lib` della cartella di installazione di BlueJ.

Per far partire il programma, basta eseguire il metodo `main` della classe `jlspr`, usando come parametro di ingresso la stringa "`nodft`" se si vuole usare il metodo basato sulle regole associative , e la stringa "`dft`" se si vuole usare il modello *LSPR*. Se il processo va a buon fine, utilizzando BlueJ nel caso del modello *LSPR* dovrebbe comparire una schermata simile a quella di figura 5.2a, mentre nel caso di utilizzo del modello basato sulle regole associative la schermata dovrebbe essere simile a quella di figura 5.2b.



(a)



(b)

Figura 5.2: Schermata di riepilogo per il modello *LSPR* (a) e per il modello basato sulle regole associative (b).

Capitolo 6

Esperimenti

In questo capitolo verranno presentati i risultati ottenuti dai due modelli di reperimento implementati in questa tesi: il modello basato sulle regole associative, che servirà come riferimento, e il modello *LSPR*. I test sono stati effettuati sulla collezione sperimentale CACM. Questa collezione comprende un insieme di 3204 documenti riguardanti l'informatica, 64 interrogazioni e i giudizi di rilevanza, cioè l'elenco dei documenti rilevanti per ogni interrogazione.

Il software, come già spiegato nel capitolo 5, è stato scritto con l'ambiente di sviluppo BlueJ, basato su Java SDK 1.6. Per quanto riguarda l'hardware, i test sono stati effettuati su un notebook con processore Intel® Pentium® M 740 funzionante ad una frequenza di 1.73 GHz, con 1.25 GB di memoria RAM e sistema operativo Microsoft® Windows® XP Home Edition con Service Pack 3.

6.1 Parametri di valutazione dell'efficacia di un sistema di reperimento dell'informazione

Nell'ambito della valutazione dell'efficacia di un sistema di reperimento dell'informazione sono state utilizzate varie misure, ma ultimamente ce n'è una che si è imposta: il MAP (*Mean Average Precision*). Prima di spiegarla e indispensabile introdurre però i concetti di richiamo e precisione.

- **Richiamo:** è il rapporto tra il numero di documenti rilevanti tra quelli reperiti e il numero di documenti rilevanti in risposta a una interrogazione.

- **Precisione:** è il rapporto tra il numero di documenti rilevanti tra quelli reperiti e il numero di documenti reperiti in risposta a una interrogazione.

Il richiamo fornisce informazioni su quanto il sistema riesce a recuperare documenti rilevanti al variare dei documenti rilevanti nella collezione, mentre la precisione dice quanto il sistema riesce a recuperare documenti rilevanti al variare dei documenti reperiti. Queste due grandezze sono inversamente legate, infatti solitamente a un aumento di richiamo corrisponde una diminuzione della precisione, e viceversa. Usare solamente richiamo e precisione per valutare l'efficacia di un sistema di reperimento dell'informazione può portare a dei problemi, ad esempio non si riesce a calcolare il richiamo se non si sa quanti sono i documenti rilevanti della collezione, così come non si riesce a calcolare la precisione se nessun documento viene reperito.

Il MAP, che è il parametro usato in questa tesi per il confronto dell'efficacia dei due modelli di reperimento dell'informazione, è la media delle precisioni medie non interpolate, essendo quest'ultima la media dei valori di precisione calcolati ad ogni documento rilevante tra quelli restituiti in risposta a una interrogazione. A livello grafico, il MAP è anche l'area sottesa dal grafico della relazione tra richiamo e precisione.[1]

6.1.1 Valutazione dei risultati: Trec Eval

Molte sono le iniziative di sperimentazione di sistemi di reperimento dell'informazione condotte a livello internazionale; la prima di queste, e probabilmente la più importante, è TREC (*Text REtrieval Conference*).[2]

Subito si rese necessario uno strumento che consentisse di calcolare l'efficacia dei sistemi di reperimento dell'informazione valutati durante i vari test: questo software si chiama Trec Eval, ed è stato usato per calcolare i vari parametri che indicano quanto un sistema è efficace.[3] Uno dei parametri calcolati da Trec Eval è proprio il MAP.

6.2 Parametri di valutazione dell'efficienza di un sistema di reperimento dell'informazione

Per valutare l'efficienza dei modelli implementati, sono stati valutati tre aspetti:

- il tempo trascorso tra l'avvio del programma e la creazione del file contenente i risultati di Trec Eval; questo tempo verrà chiamato successivamente *tempo di esecuzione*;

[1] Per approfondimenti consultare [1].

[2] Maggiori informazioni si trovano all'URL `http://trec.nist.gov/` e in [1].

[3] Il software è disponibile all' URL `http://ims.dei.unipd.it/websites/archive/ims2009/members/agosti/teaching/2007-08/ir/docs-per-studenti/trec_eval/`, anche se è comunque già presente all'interno del directory `trec_eval` del progetto, come spiegato nel capitolo 5. Una guida sulla preparazione dei dati nel formato corretto per Trec Eval è presente in [9].

- l'occupazione di memoria principale;
- le dimensioni su disco dei file creati durante l'elaborazione (quelli descritti nel capitolo 5).

Sebbene lo scopo sia stato più l'efficacia dell'efficienza, e che quindi questi parametri possano sicuramente essere migliorati, essi sono stati calcolati per dimostrare che comunque, almeno per collezioni non molto grosse in termini di quantità di dati, l'efficienza è accettabile.

6.3 Risultati

Di seguito verranno presentati i risultati ottenuti con le implementazioni dei due modelli (quello basato sulle regole associative, o ARS, e *LSPR*), sia riguardo all'efficacia sia all'efficienza. Per quanto concerne il tempo di esecuzione e l'occupazione in memoria, i valori riportati sono i valori medi dopo 10 esecuzioni del programma.

Modello	**MAP**	**Tempo di esecuzione [s]**	**Occupazione di memoria [MB]**	**Occupazione su disco [MB]**
ARS	0.2425	42.182	205	5.45
LSPR	0.3476	52.875	220	5.45

Tabella 6.1: Prestazioni del modello basato su regole associative e del modello *LSPR*.

Come si può notare, l'efficacia di *LSPR* è molto maggiore rispetto a quella del modello basato sulle regole associative: l'aumento del valore di MAP è del 43.3% circa. Il prezzo che si paga è però nella diminuzione dell'efficienza, in quanto l'utilizzo della DFT fa aumentare il tempo di esecuzione del 25.3% circa (anche se presumibilmente in questo caso ha più senso considerare l'aumento assoluto in secondi che quello percentuale, dato che la maggior parte del tempo riguarda le operazioni di indicizzazione e di calcolo delle regole associative comuni ad entrambi i modelli), e l'occupazione di memoria del 7.3% circa.

Una cosa interessante sarebbe il confronto tra il MAP di *LSPR* e quello di altri sistemi di reperimento dell'informazione, per capire dove il modello sviluppato in questa tesi si colloca.

Durante il corso di reperimento dell'informazione tenuto dalla professoressa Maristella Agosti nell'anno accademico 2007/2008, è stato svolto da vari gruppi un progetto che consisteva nel valutare le prestazioni di alcuni

Software	**Versione**	**MAP**
Copernic Desktop Search	2	0.0074
Copernic Desktop Search	2.3	0.0110
Strigi	0.5.7	0.0153
Meta Tracker	0.6.4	0.0208
Recoll	1.10.1	0.2769
Xapian	1.0.5	0.3201
Terrier	2	0.3240
Desktop Terrier	1.1	0.3287

Tabella 6.2: MAP di alcuni software su collezione CACM

software scaricabili da internet sulla collezione CACM.[4] I valori di MAP ottenuti sono riportati nella tabella 6.2.[5]

Come si può notare, il software basato sul modello *LSPR* ha un valore di MAP superiore a quello di tutti gli altri programmi valutati, e superiore del 5.7% circa rispetto al miglior risultato presente in tabella 6.2, cioè quello di Desktop Terrier.

[4]Tale corso è uno dei corsi della Laurea Specialistica in Ingegneria Informatica presso l'Università degli Studi di Padova.

[5]Si può trovare una documentazione più accurata riguardo ai risultati ottenuti dai vari software all'URL `http://ims.dei.unipd.it/websites/archive/ims2009/members/agosti/teaching/2007-08/ir/sitoWebIR/index.htm`

Capitolo 7

Conclusioni

Il modello sviluppato in questa tesi di laurea ha dimostrato che l'utilizzo della DFT nell'ambito del reperimento dell'informazione può garantire buoni livelli di efficacia (soprattutto grazie al fatto di aver sfruttato come un vantaggio l'effetto di dispersione spettrale, che di solito è una caratteristica negativa nell'elaborazione numerica dei segnali). Infatti, come si può osservare dalla tabella 6.1, l'aumento del valore del MAP rispetto al modello di riferimento è del 43.3% circa, ed è migliore di quello ottenuto dai vari software riportati in tabella 6.2.

Sono molti però i parametri che possono essere variati, ad esempio il numero di punti da assegnare a ogni termine dell'interrogazione nello spettro, il tipo di filtro usato e la sua selettività, lo schema di pesatura sia dei termini della collezione sia di quelli dell'interrogazione, i parametri e la soglia dell'attendibilità, il metodo di assegnamento delle frequenze ai termini (si può in questo senso pensare di usare anche vocabolari di sinonimi in modo da assegnare frequenze vicine a termini sinonimi, magari anche di lingue diverse per poter supportare il *cross-language information retrieval*), e ancora altri parametri si cui si è discusso nel corso della tesi. In particolare per quanto riguarda la selettività, si è visto che per valori maggiori e minori di 24 il MAP diminuisce, come se questo rappresentasse un punto si massimo di una qualche funzione che lega MAP a selettività; sarebbe interessante capirne il motivo, e magari riuscire a determinare questa funzione, se esiste.

Durante l'implementazione del modello non si è speso troppo tempo per la ricerca dei parametri ottimi, che consentissero di ottenere il MAP migliore: questo significa che un lavoro in questo senso potrebbe probabilmente portare a un aumento del MAP ottenuto dal modello, in quanto è poco realistico pensare di aver trovato la configurazione ottima dopo poche prove.

Un lavoro importante da fare riguarda anche l'aumento dell'efficienza di questo modello. Come detto più volte, questo non era uno degli obiettivi principali di questa tesi, ma visto che l'efficacia ottenuta è buona, ha senso

investire nell'ottimizzazione del codice per ridurre il tempo di esecuzione e l'occupazione sia di memoria centrale che di memoria secondaria. Per quanto riguarda il tempo di esecuzione, è stata fatta un'analisi approssimativa delle complessità delle varie parti del progetto, in modo da evidenziare i punti deboli e fornire indicazioni utili per una successiva revisione del codice. Non si è deciso di analizzare l'occupazione di memoria centrale e secondaria, perchè l'utilizzo di strutture dati più efficienti in termini di tempo di calcolo può portare ad una riduzione della memoria necessaria, e comunque le scelte di ottimizzazione dell'occupazione della memoria dipendono dalle scelte delle strutture dati usate per migliorare il tempo di esecuzione. Un semplice accorgimento che potrebbe far aumentare sia efficacia sia efficienza sarebbe quello di eliminare dai risultati quei documenti che presentano una potenza superiore a una certa soglia: questo ridurrebbe le dimensioni del file `risultati.txt`, e probabilmente aumenterebbe il MAP.

Verrà di seguito presentata l'analisi dei tempi di esecuzione, in notazione asintotica, per le varie parti che compongono il modello *LSPR*, dove sono state considerate operazioni con costo $O(1)$ le operazioni aritmetiche e i confronti.[1] Bisogna introdurre però quattro simboli nuovi: SW rappresenta l'elenco delle stop word, T_c rappresenta l'insieme dei termini distinti della collezione, T_q rappresenta l'insieme dei termini distinti della query e T rappresenta il numero di termini totali della collezione, cioè $T = \sum_j X_j$.

- **Rimozione delle stop word dalla collezione:** per ogni termine della collezione, esso viene confrontato con la lista delle stop word; essendo quest'ultima ordinata alfabeticamente, si può fare una ricerca logaritmica; la complessità che si ottiene è allora $T_{sw} \in O\left(|T| \cdot \log |SW|\right)$.

- **Stemming sulla collezione:** il tempo di esecuzione dipende dall'algoritmo di Porter, e dato che è stato utilizzato un pacchetto che lo implementava, non si può determinare precisamente la complessità, che dipenderà comunque dal numero di termini della collezione $|T|$.

- **Matrice termini - documenti:** il costo per la creazione di questa matrice è più un costo di I/O che non di work fatto in memoria centrale. Durante lo svolgimento della tesi sono state create però due strutture, utili a velocizzare le operazioni successive: una lista di documenti a cui corrisponde l'elenco dei termini (o meglio il numero di ripetizioni) che contengono (chiamata in seguito DT), e una lista di termini contenente la lista dei documenti dove sono presenti (chiamata in seguito TD). Questo accorgimento permette di superare il limite introdotto dalla grande sparsità della matrice termini - documenti. Per creare queste due strutture, è stato necessario creare l'elenco T dei termini distinti della collezione: per fare questo si è dovuto ordinare l'elenco dei termini

[1] Maggiori informazioni sull'utilizzo della notazione asintotica nell'analisi degli algoritmi possono essere trovate in [2].

della collezione e poi rimuovere i duplicati.[2] La complessità ottenuta è $T_{t_c} \in O(|T| \cdot \log|T| + |T|) = O(|T| \cdot \log|T|)$.

- **Pesatura dei termini della collezione**: Per quanto riguarda la pesatura TF, usando la struttura DT la complessità risulta $T_{tf} \in O(|C| \cdot |T_c|)$ per quanto riguarda la pesatura IDF, si è sfruttata la struttura TD, ottenendo una complessità $T_{idf} \in O(|T_c|)$. La normalizzazione non aumenta la complessità asintotica, quindi la pesatura TF-IDF costa in totale $T_{tf-idf} \in O(|C| \cdot |T_c|)$.

- **Rimozione delle stop word dalla query:** con considerazioni simili a quelle fatte per la rimozione delle stop word dalla collezione, si ottiene una complessità $T_{swq} \in O(|Q| \cdot \log|SW|)$.

- **Stemming sull'interrogazione:** in modo simile a quanto detto per la collezione, lo stemming dipende dall'algoritmo di Porter, e dipenderà dal numero di termini dell'interrogazione $|Q|$.

- **Ordinamento e rimozione dei duplicati di una query:** per ottenere T_q, bisogna ordinare i termini dell'interrogazione, usando il metodo sort() della classe Arrays di Java, e poi dimuovere i duplicati. La complessità è $T_{t_q} \in O(|Q| \cdot \log|Q| + |Q|) = O(|Q| \cdot \log|Q|)$.

- **Pesatura IDF dell'interrogazione:** ogni termine dell'interrogazione è stato pesato con la pesatura IDF, in maniera simile a quanto fatto per i documenti. Se un termine dell'interrogazione non è presente in T_c (si può verificarlo con una ricerca dicotomica), allora il peso di quel termine è 0; altrimenti il peso è il logaritmo del rapporto tra il numero di documenti della collezione e il numero di documenti che contengono quel termine. La complessità risultante è $T_{idfq} \in O(|T_q| \cdot \log|T_c|)$.

- **Calcolo delle regole associative** (per una interrogazione)**:** data una interrogazione Q, bisogna calcolare supporto e confidenza delle regole associative dove il primo termine appartiene all'interrogazione e il secondo alla collezione; la complessità risultante è $T_{ar} \in O(|T_q| \cdot |T_c|)$. Questa fase deve essere ripetuta per ogni interrogazione.

- **Calcolo dei gruppi associativi** (per una interrogazione)**:** data una interrogazione Q, bisogna scandire le regole associative e considerare solo quelle con attendibilità superiore alla soglia stabilita. La complessità risulta quindi $T_{arg} \in O(|T_q| \cdot |T_c|)$. Questa fase deve essere ripetuta per ogni interrogazione.

[2] Si è usato il metodo sort() della classe Arrays di Java, che implementa l'algoritmo QuickSort, ma l'elenco dei termini era troppo lungo e il metodo restituiva l'elenco ordinato troncato. Si è quindi deciso di spezzare in due l'elenco, ordinarlo separatamente e poi fonderlo con una fase di merge.

- **Calcolo della collezione ridotta:** Per ogni termine dell'interrogazione bisogna trovare i documenti che contengono almeno un termine del gruppo associativo relativo al termine dell'interrogazione analizzato. Ognuno dei $|T_q|$ gruppi associativi ha al massimo $|T_c|$ termini, e usando la struttura TD si può cercare in tempo logaritmico l'elenco dei documenti associati a un termine. Visto che i gruppi associativi sono stati calcolati precedentemente, la complessità risulta $T_{cr} \in O(|T_q| \cdot |T_c| \cdot \log |T_c|)$, anche se questa è una stima per eccesso, perchè al posto del prodotto $|T_q| \cdot |T_c|$ si può mettere la cardinalità totale di tutti i gruppi associativi.

- **Calcolo dello spettro:** il calcolo delle frequenze ha complessità $O(|T_q|)$, il calcolo dei campioni ha complessità $O(N \cdot |T_q|)$ e il calcolo della DFT ha complessità $O(N \log N)$. Dato che per costruzione si ha che $|T_q| \ll |Q| \ll N$, come si può notare dall'equazione (4.3), allora la complessità risulta $T_{spect} \in O(N \log N)$.

- **Filtraggio:** supponendo che lo spettro sia già stato calcolato, il filtraggio si può pensare come una moltiplicazione dei punti dello spettro per valori che vanno da 0 a 1; la complessità è dunque $T_{fil} \in O(N)$.

- **Ranking:** dopo il calcolo della collezione ridotta (già eseguito in precedenza), che ha una dimensione sicuramente $O(|C|)$, per ogni documento della collezione ridotta bisogna applicare l'algoritmo di filtraggio, che ha complessità $O(N)$, allo spettro dell'interrogazione calcolato precedentemente. Il calcolo della potenza ha complessità $O(N)$, e l'ordinamento dei documenti per potenza crescente, sebbene nel progetto sia stato eseguito con l'algoritmo Bubblesort (avente complessità $O(n^2)$ se n è la dimensione dell'input), può essere calcolato in $O(|C| \cdot \log |C|)$, dato che i documenti da ordinare sono $O(|C|)$. La complessità totale è allora $T_{rank} \in O(N + |C| \cdot \log |C|)$. Anche questo deve essere ripetuto per ogni interrogazione.

Una volta aumentata l'efficienza attraverso una revisione del codice, e trovati dei buoni parametri che aumentino anche l'efficacia, sarebbe interessante provare a vedere come il modello funziona su altre collezioni sperimentali, comprendenti collezioni più grandi sia in termini di numero di documenti che di occupazione su disco. Se le prestazioni si confermassero buone, si potrebbe pensare di utilizzare questo modello in altri ambiti, come per i motori di ricerca per il Web e nelle reti P2P, o provare a combinarlo con altri modelli già esistenti.

Bibliografia

[1] Agosti, Maristella e Massimo Melucci: *Reperimento dell'informazione: concetti, architetture e modelli dei motori di ricerca*, 2007.

[2] Cormen, T. H., C. E. Leiserson, R. L. Rivest e C. Stein: *Introduzione agli algoritmi e strutture dati.* McGraw-Hill, seconda edizione, 2005.

[3] Mian, Gian Antonio: *Elaborazione numerica dei segnali.* URL: `http://lttm.dei.unipd.it/nuovo/teaching/ens/HTML/dispense.html`, 2006.

[4] Narduzzi, Claudio: *Dispense di Misure Elettroniche*, 2006.

[5] Oppenheim, Alan V., Alan S. Willsky e S. Hamid Nawab: *Signals & systems (2nd ed.).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[6] Papapetrou, Odysseas: *Full-text indexing and information retrieval in p2p systems.* Nel *Ph.D. '08: Proceedings of the 2008 EDBT Ph.D. workshop*, pagine 49–57, New York, NY, USA, 2008. ACM.

[7] Press, William H., Saul A. Teukolsky, William T. Vetterling e Brian P. Flannery: *Numerical recipes in C (2nd ed.): the art of scientific computing.* Cambridge University Press, 1992.

[8] Tan, Pang Ning, Michael Steinbach e Vipin Kumar: *Introduction to Data Mining, (First Edition).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[9] Zorzan, Emmanuele: *Linee guida per l'utilizzo del software Trec_eval.* URL: `http://ims.dei.unipd.it/websites/archive/ims2009/members/agosti/teaching/2007-08/ir/docs-per-studenti/LineeGuidaTrec_eval_Zorzan-2008.pdf`, 2008.