Università degli studi di Padova
Facoltà di Ingegneria
Dipartimento di ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Induction Team

Progetto Amerigo Vespucci

Analisi del software di desktop search Recoll
Corso di reperimento dell'infomrazione
Prof. Maristella Agosti

Data Inizio: 07 Febbraio 2007
Data Conclusione: 26 Marzo 2007

Componenti del Team:

Dindo Stefano	566949-IF
Sandri Francesco	566520-IF
Scremin Thomas	566991-IF
Tono Raffaella	569864-IF
Vantini Marco	566948-IF

Introduzione

Recoll è un software di tipo *desktop search*, ossia un'applicazione che con un'operazione di *data minig* consente di estrarre informazioni dai file presenti in un computer (indicizzazione) per poi permettere all'utente di effettuare ricerche (reperimento) di parole o termini, basandosi sul *contenuto* dei file e non sul nome o l'estensione degli stessi.
Recoll è un'applicazione di tipo Open Source con licenza GPL. La versione utilizzata è la 1.7.5, disponibile per le seguenti distribuzioni di Linux: FreeBSD, Darwin e Solaris (versioni FreeBSD 5.5, Redhat 7.3, Fedora Core 5, Suse 10.1, Gentoo, Debian 3.1 e Solaris 8/9).L'interfaccia utente è fornita mediante l'ausilio di **QT** (http://ftp.iasi.roedu.net/mirrors/ftp.trolltech.com/qt/source/qt-x11-opensource-src-4.2.2.tar.gz), una piattaforma di sviluppo già presente dalla distribuzione Fedora Core 5. Un altro componente essenziale è la libreria di ricerca ***Xapian,*** che si basa su un modello di rilevanza probabilistico con pesatura per l'indicizzazione dei documenti (il metodo di pesatura è riportato a pag. 13); essa memorizza le informazioni in formato UNICODE UTF a 8 bit.
Attraverso il file di configurazione *recoll.conf* è possibile impostare numerose personalizzazioni, come:

- modalità di indicizzazione;
- elenco delle directory e/o file da sottoporre all'indicizzazione e tra queste eventuali sottodirectory da escludere;
- formato dei file da considerare (purché solo testo, non multimediali);
- scelta della lingua per l'espansione grammaticale delle parole;
  (specificando più lingue vengono creati database distinti);
- set di caratteri da considerare di default;
- dimensione del riassunto. Mediante interfaccia, invece, oltre a varie modalità di visualizzazione dei risultati, si può:
- specificare una ricerca per "raffinamento" (ossia man mano che si digitano le parole nella casella di ricerca i documenti vengono filtrati scartando man mano quelli meno rilevanti);
- abilitare la costruzione del riassunto di un documento, decidendone la dimensione.

L'indicizzazione al primo avvio è automatica; successivamente è possibile scegliere tra quella periodica, (lanciando il programma *recollindex*) oppure quella in tempo reale, che rileva le variazioni intervenute sui file (modifica o creazione). L'utente eventualmente può adottare entrambe le modalità purché specificate su directory distinte. L'applicazione non memorizza copia dei file indicizzati, ma crea un surrogato che permette comunque una ricostruzione comprensibile del testo originale.
Il risultato di una ricerca è una lista di documenti contenenti la parola o i termini richiesti in ordine di rilevanza decrescente, secondo una stima effettuata dal sistema. Dato che le esigenze di ricerca spesso non coinvolgono una singola parola ma anche sue declinazioni, (derivati, plurali, verbi coniugati o parole ottenute da una radice comune) questo software estrae la radice linguistica di una parola ed estende la ricerca tenendo conto delle diverse varianti (Descrizione algoritmo di stemming a pag 11). Lo stemming può essere disabilitato scrivendo in maiuscolo il primo carattere; se presenti in altre posizioni, maiuscole o minuscole non hanno alcuna influenza sulla ricerca. Altre modalità di ricerca prevedono l'uso di caratteri jolly e le espressioni regolari; installando un componente software aggiuntivo, consente anche la ricerca fonetica basata sulla pronuncia (non testata).Anche se non specificato nella documentazione, con dei test si è potuto verificare che *Recoll* non è in grado di gestire la sinonimia dei termini e nemmeno la polisemia.
Tra le funzionalità non previste si segnala la rimozione delle stop word (per la quale è stato necessario creare apposite classi in java) e il riconoscimento automatico della lingua.

Installazione

Recoll è un software per ambiente Linux e Unix, pertanto, come la maggior parte dei software dedicati a queste piattaforme, può essere installato tramite:
- pacchetti pre-compilati;
- compilazione da sorgente.

Tenuto conto delle dipendenze software già specificate, è consigliabile installare prima i pacchetti *Xapian* (http://www.oligarchy.co.uk/xapian/0.9.9/xapian-core-0.9.9.tar.gz) e *Libiconv* (http://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.11.tar.gz) e procedere successivamente con l'installazione di *Recoll*.

La modalità da noi seguita è stata quella tramite pacchetto pre-compilato in quanto riduce sensibilmente i tempi di installazione rispetto alla compilazione da sorgente. Una volta scaricato il pacchetto *recoll-1.7.5-1.i386.rpm* (http://www.lesbonscomptes.com/recoll/recoll-1.7.5-1.i386.rpm) si avvia una semplice procedura guidata di installazione mediante doppio click sul file scaricato. A questo punto l'installazione base di *Recoll* è completata, tenendo conto tuttavia che, per poter indicizzare alcuni tipi di file (per es. PDF, MS Word, MS Excel, Postscript, etc.), è necessario installare ulteriori moduli specifici. Dato che la collezione a disposizione prevede solo file di tipo *txt* tali componenti aggiuntivi non sono stati installati.

Al primo avvio, *Recoll* esegue, di default, la prima indicizzazione automatica dei file dell'intero disco (/home) a meno che non si intervenga sul file di configurazione *recoll.conf*. All'interno di questo file ci sono tre tipi di linee:
- *commenti:* righe che iniziano con #;
- *settaggio dei parametri:* che prevedono un'assegnazione (*nome = valore)*;
- *definizione delle sezioni:* in cui specificare le directory.

Alcuni parametri utili in fase di configurazione sono:
- ***topdirs***: si specificano le cartelle e/o i file da indicizzare, tenendo presente che il software ignora qualsiasi collegamento simbolico;
- ***dbdir***: nome della cartella *Xapian* dei dati;
- ***skippedNames***: si indica una lista di cartelle e nomi di file, separati da uno spazio vuoto, di quello che non si vuole venga indicizzato;
- ***filtersdir***: indica la cartella nella quale inserire gli *script* esterni necessari a indicizzare alcuni tipi di file;
- ***indexstemminglanguages***: una lista di lingue per le quali verrà costruito il database per l'espansione grammaticale;
- ***indexallfilenames***: attraverso questo parametro si decide se indicizzare i nomi di file in modo indipendente o meno dal suffisso. È utile perché viene permesso di ricercare nomi anche usando i caratteri jolly;
- ***idxabsmlen***: definisce la dimensione del riassunto di ogni file indicizzato (il valore di default è di 250 caratteri). Al termine della ricerca, *Recoll* mostra nella lista dei risultati trovati, i riassunti, senza dover leggere il documento originale.

Analisi dei risultati

Il team ha eseguito numerosi test sottoponendo a *Recoll* diverse interrogazioni per valutare l'efficacia dello strumento di reperimento scelto. Tali risultati sono stati poi elaborati mediante il software di valutazione messo a disposizione dal docente: *trec_eval*, versione 8.1.
Come prima cosa, si è reso necessario creare degli applicativi in Java (allegati in appendice a pagina 15) sia per separare i testi presenti all'interno del singolo file *Time.txt* e sia per rimuovere le stop word da un file di testo mandato in ingresso.
Per la natura dello strumento preso in esame, non si è potuto automatizzare in tutto o in parte la fase di input delle interrogazioni al software, e quindi abbiamo dovuto inserire manualmente ogni query e prelevarne i risultati per poi formattarli (tramite strategici programmi in Java riportati da pagina 13 e successive) secondo le specifiche fornite per poter essere mandati in ingresso al programma *trec_eval*.
Le valutazioni che abbiamo ritenuto opportuno compiere sono state:

- valutazione dello strumento con funzionamento di default (senza alcuna personalizzazione o modifica);
- valutazione dello strumento escludendo l'operazione di stemming;
- valutazione dello strumento eliminando le stop word dalla collezione e dalle query, sia con stemming che senza stemming.

Il primo test prevedeva l'uso dell'operatore *AND* tra le parole delle query. Questa scelta tuttavia si è rivelata inadeguata poiché il numero delle query restituite era molto basso e quindi la stima delle prestazioni con *trec_eval* riportava valori non significativi (0.0000 in tutti i campi). Per questo motivo abbiamo preferito proseguire la valutazione con l'operatore *OR*.
Per compiere questi confronti abbiamo deciso di limitare il numero di query a 21 (prendendone una ogni quattro) fatta eccezione per il caso di funzionamento normale del software per il quale abbiamo compiuto una doppia valutazione: una con tutte le 83 query (i cui risultati sono stati poi inseriti nella tabella delle prestazioni) ed una con solo 21 query per compiere un confronto con gli altri casi presi in esame.
Il primo problema che il nostro gruppo ha dovuto affrontare è stato la presenza di valori, risultanti dalla valutazione con *trec_eval*, eccessivamente bassi (e.g. MAP=0.0123, P5=0.049, etc.). Valori che ad un primo confronto mettevano in luce come lo strumento funzionasse in modo peggiore con la rimozione delle stop word rispetto al loro mantenimento (vedi grafico 5 richiamo precisione a pag. 7). Si è successivamente scoperto che la collezione presa in esame era errata e per questo motivo è stata sostituita.
Con il cambiamento della collezione, le interrogazioni con l'operatore *AND* hanno prodotto risultati osservabili ma nuovamente troppo bassi per permettere il riconoscimento delle prestazioni effettive (e.g. MAP=0.2956, R-Precisione=0.3372, etc.) (vedi grafico 4 pag 7). Ancora una volta, quindi, abbiamo optato per l'*OR*.
La valutazione sulla collezione esatta ha dato i risultati attesi dai quali si può evincere il reale funzionamento del sistema.
Analizzando i risultati così ottenuti si può desumere che lo strumento funziona abbastanza bene con un valore di **MAP=0.6449**, **R-Precisione=0.5875**, **P5=0.388**, **R5=0.6224**, **P10=0.2819**, **R10=0.7963** i quali si dimostrano essere relativamente alti rispetto alla media (vedi grafici pag. 5 e 6 e tabella a pagina 8. Mentre per i risultati forniti da trec_eval vedere pag 18 e successive).
Abbiamo notato quindi che il sistema dimostra un funzionamento migliore con l'ausilio di eliminazione delle stop word e con l'azione di stemming delle parole, rispetto alla mancanza di queste operazioni (come la teoria del reperimento ci indica).

Conclusioni

Ad un primo utilizzo la personalizzazione risulta intuitiva, fatta eccezione per il cambiamento della lingua di interfaccia (che comporta una discreta conoscenza del sistema operativo Linux) e l'impostazione del formato di resa dei risultati. Tale impostazione avviene conoscendo a priori gli identificativi per gli attributi dei documenti restituiti, da inserire manualmente, consultando la documentazione, secondo una rigida sintassi. (Vedi figura 1 pag. 9)

Per quanto riguarda l'interfaccia di interrogazione standard, essa è di facile utilizzo in quanto non prevede la conoscenza di operatori booleani (tipici di un sistema di reperimento) bensì tali impostazioni vengono scelte mediante dei menù a tendina in linguaggio naturale (Vedi Figura 2 pag. 9).
D'altro canto lo strumento mette a disposizione una modalità di ricerca avanzata il cui funzionamento risulta più complesso: richiede delle nozioni basilari del reperimento note solo a utenti esperti (per esempio la differenza tra parola e termine) (Vedi Figura 3 pagina 10).

La modalità di indicizzazione manuale prevede l'utilizzo di un comando per azionare l'indicizzazione di tutti i documenti su disco. Questo approccio richiede di lanciare questa istruzione ogni qualvolta venga creato o inserito un nuovo documento, altrimenti un'eventuale ricerca non produrrà risultati contenenti il nuovo documento. Per questo motivo tale modalità è sconsigliata ad utenti con scarse conoscenze nell'uso di questo sistema.
La modalità di indicizzazione automatica è attivabile mediante l'esecuzione del comando *recollindex -m* che tuttavia deve essere lanciato ad ogni avvio del sistema. L'esecuzione ad ogni avvio, di tale istruzione, viene resa possibile facendo eseguire al sistema uno script, soluzione sicuramente poco consigliata ad utenti non esperti di sistemi Unix.

Altri parametri relativi al funzionamento del sistema *Recoll* sono personalizzabili solo agendo direttamente sul file di configurazione *recoll.conf* rispettando un'opportuna sintassi che, nonostante sia specificata nel manuale, risulta poco intuitiva all'utente.

Questo strumento gestisce solamente file in formato testuale e non di tipo multimediale. Per l'indicizzazione di file testuali di formati diversi dal semplice *txt* si devono applicare delle estensioni reperibili dal sito del produttore e di facile installazione.

Dal punto di vista dell'efficienza, il sistema risulta rapido nel compiere l'indicizzazione e nella visualizzazione dei risultati. Tuttavia si potrebbe muovere una critica nei confronti della presentazione degli stessi in quanto, se limitata ad una singola cartella, il numero totale dei documenti reperiti è riferito a quello complessivo dei documenti presenti su disco.

Grafico Precisione Richiamo

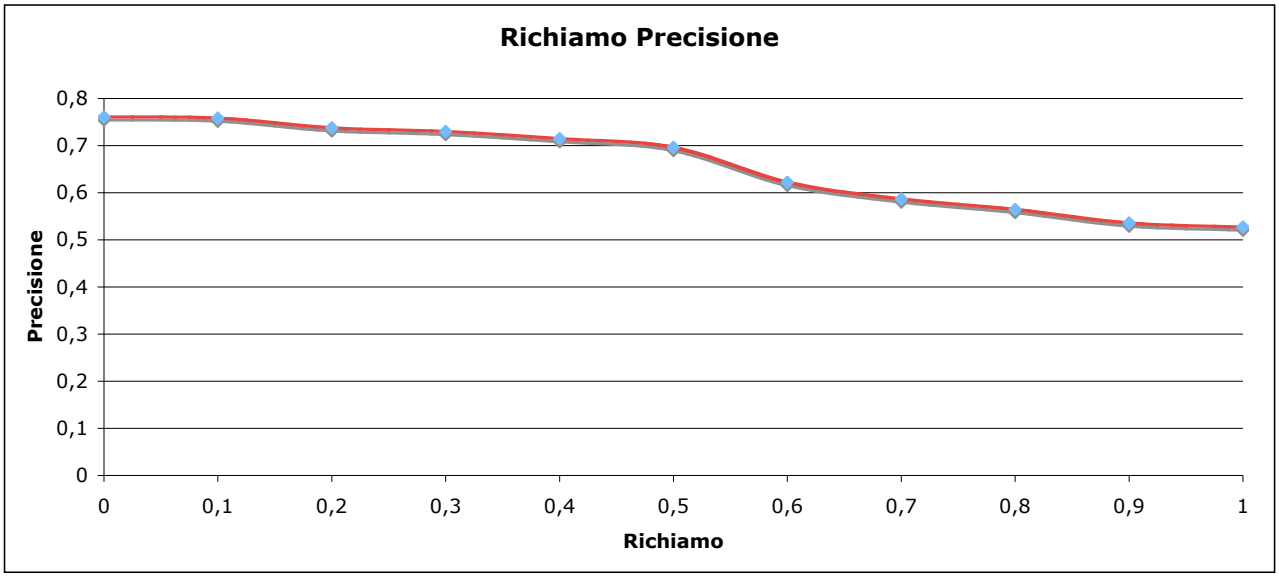

Grafico1. Grafico richiamo precisione con collezione di prova corretta

Grafico Precisione Richiamo

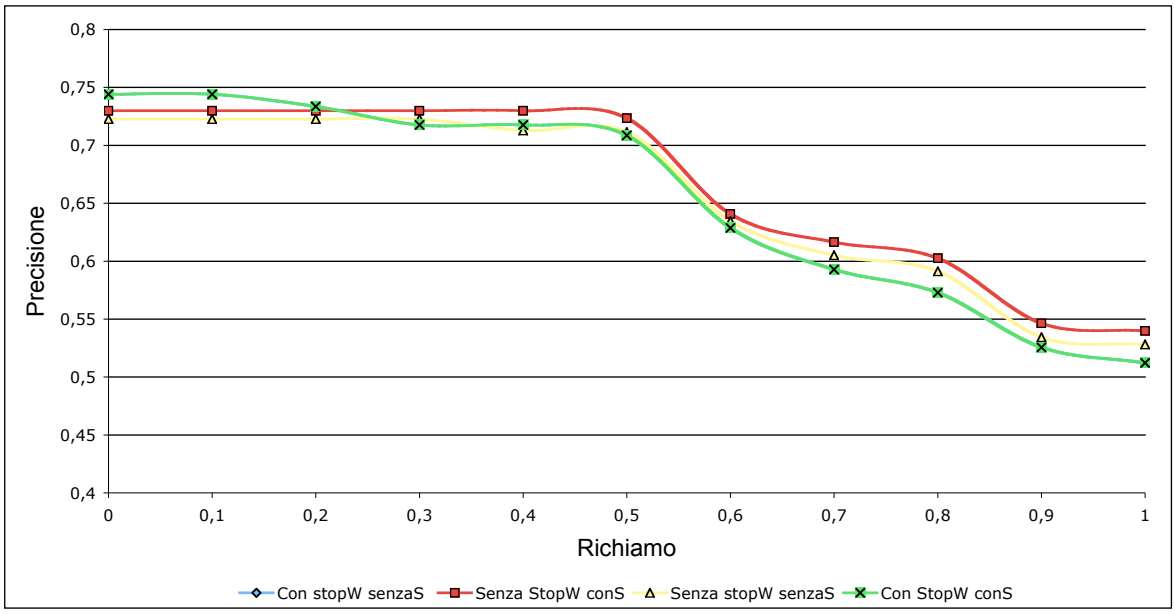

Grafico 2. Grafico precisione richiamo con tutti i casi

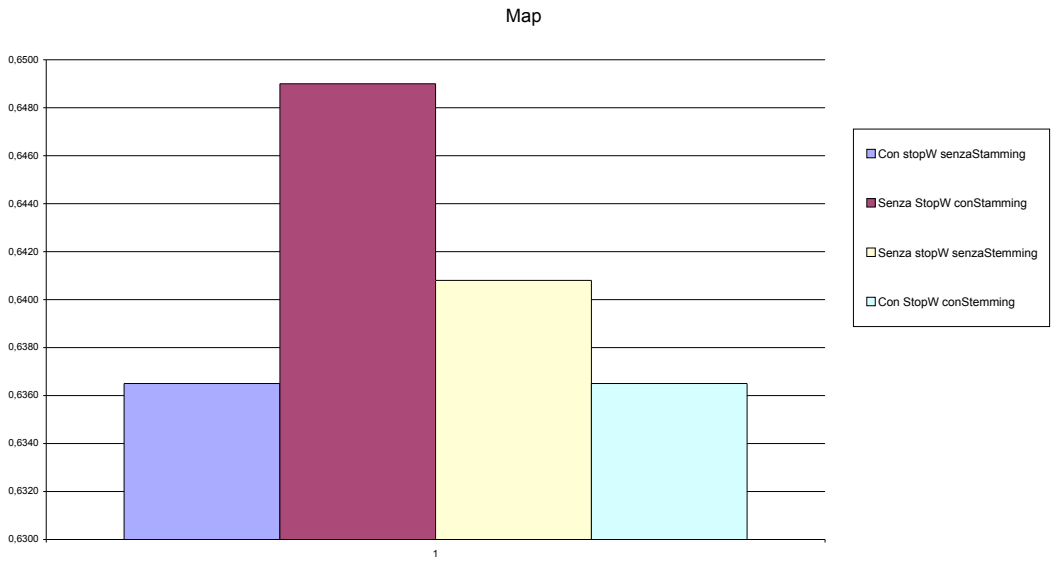

Grafico 3. Confronto dei map nelle varie configurazioni

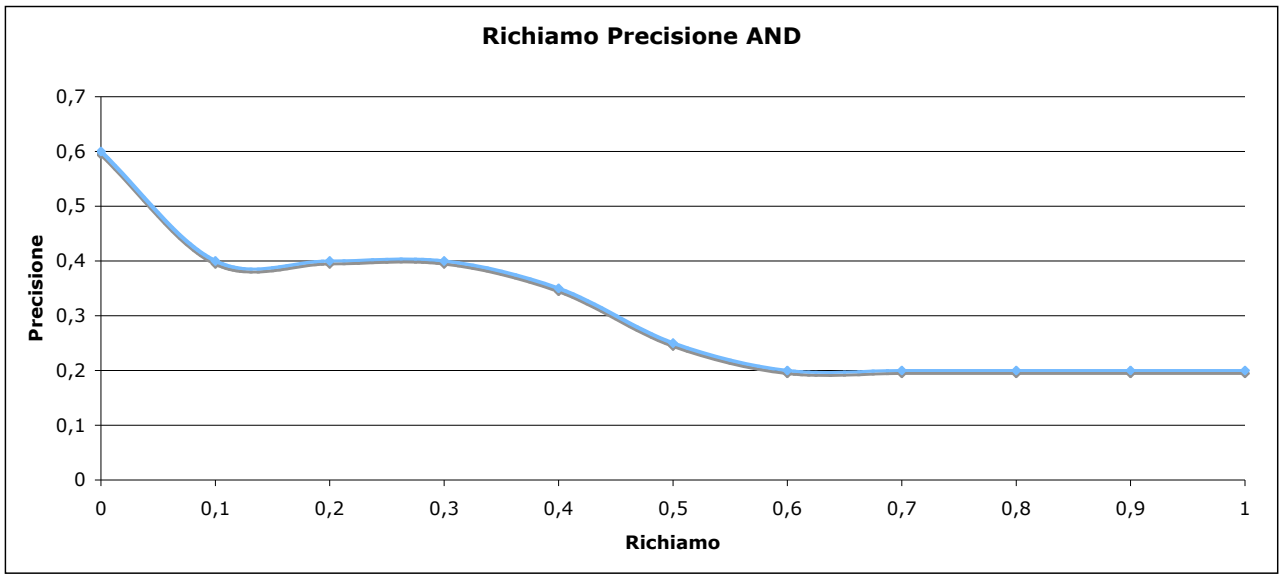

Grafico 4. Richiamo Precisione AND

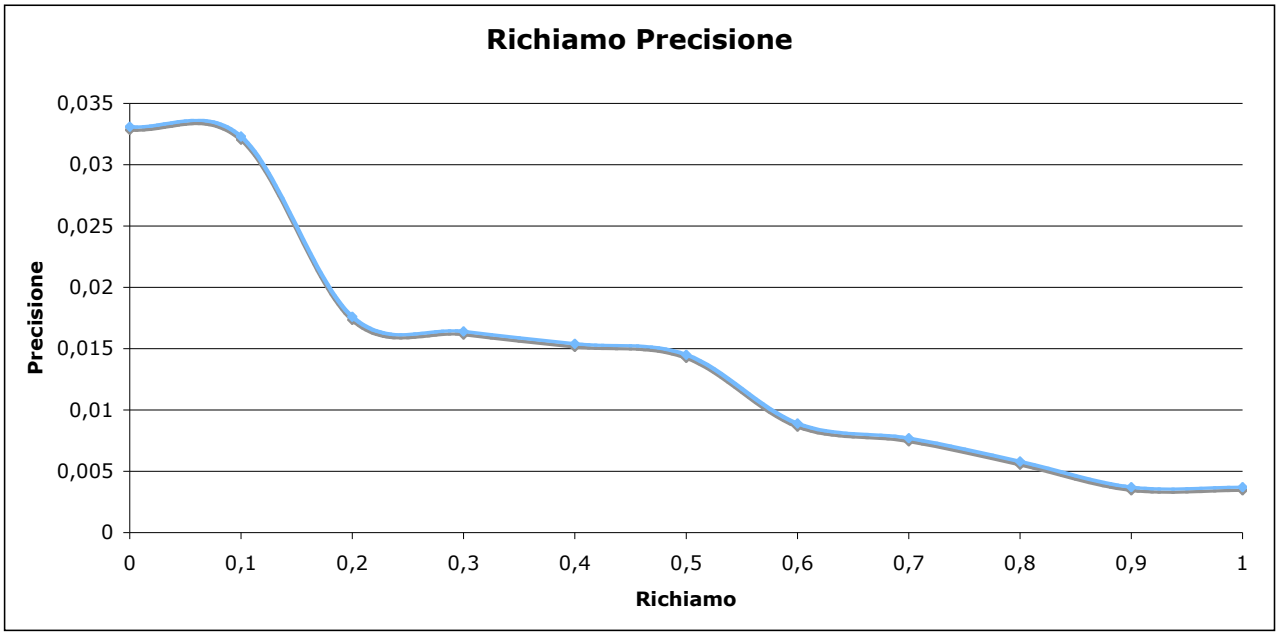

Grafico 5. Grafico richiamo precisione con collezione di prova errata.

Tabella comparativa prestazioni

Operatore	Stemming	Rimozione stop word	MAP	R-Precisione	P5	R5	P10	R10
AND	Sì	Sì	0.2956	0.3372	0.16	0.3372	0.08	0.3372
OR	Sì	Sì	0.649	0.5923	0.3905	0.6431	0.3238	0.8458
OR	Sì	No	0.6365	0.5585	0.4	0.651	0.3286	0.8537
OR	No	Sì	0.6408	0.5718	0.3810	0.6374	0.3143	0.8333
OR	No	No	0.6365	0.5585	0.4	0.651	0.3286	0.8537

Tabella dei risultati comparativi, valutati in un sottoinsieme di 21 query su 83 query

Figure 1/2

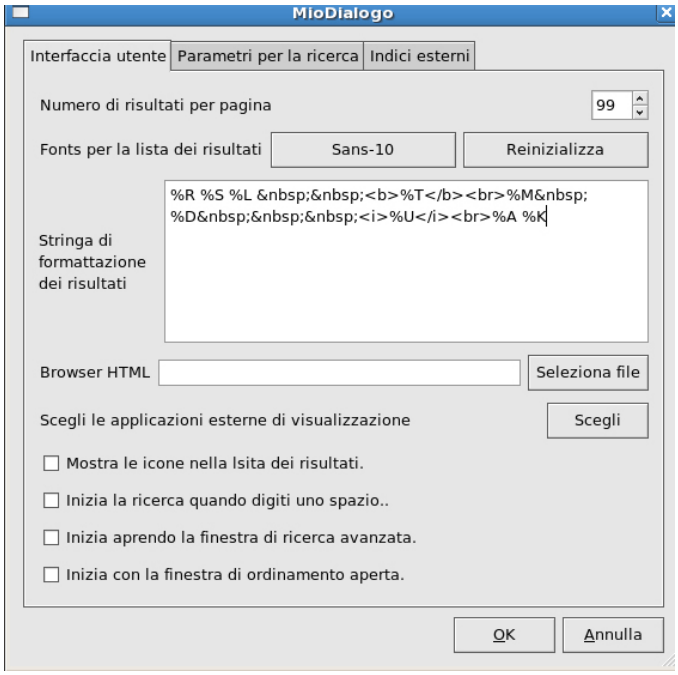

Figura 1

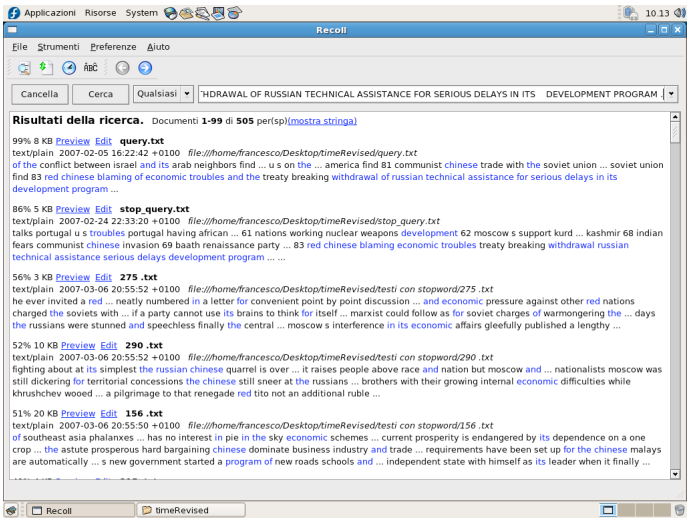

Figura 2

Figure 2/2

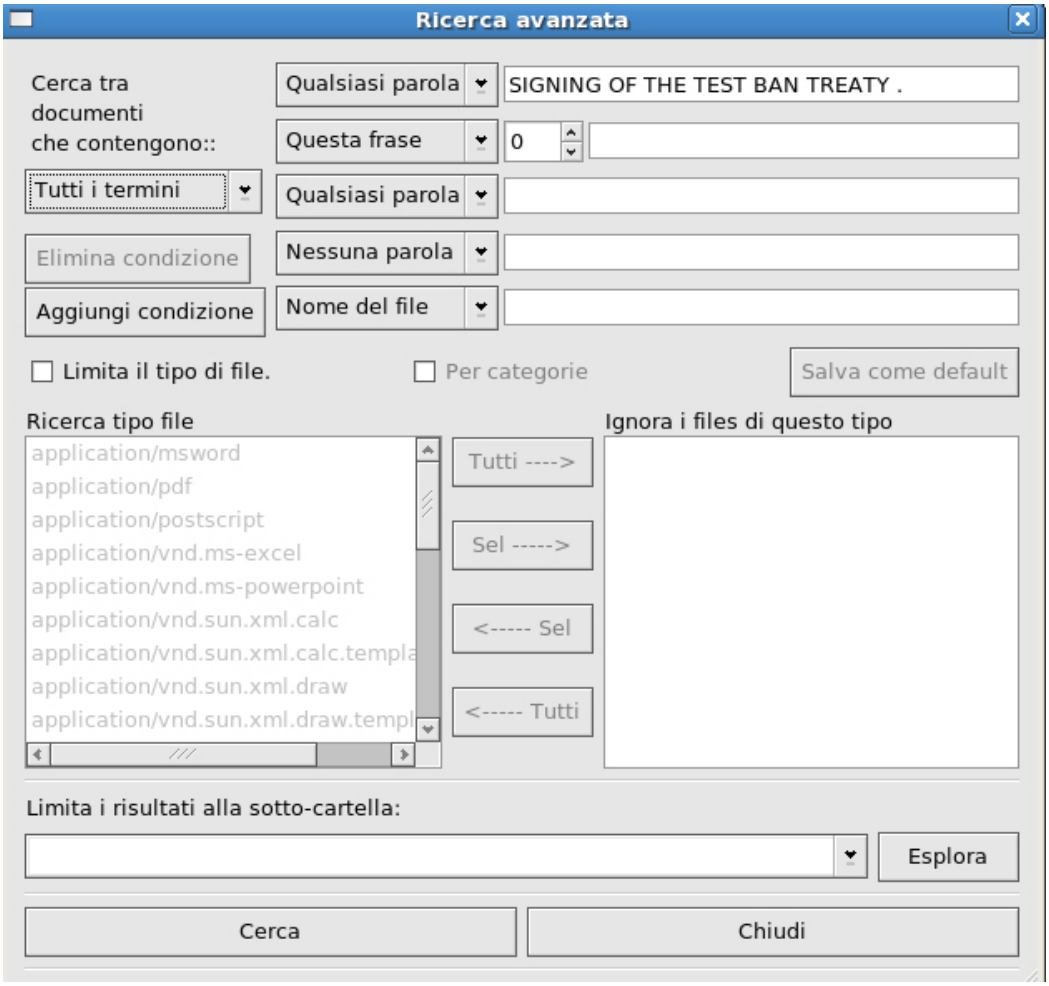

Figura 3

Funzionamento algoritmo di stemming

Definiamo una parola come la concatenazione di una o più consonanti seguita da una o più vocali; è quindi possibile schematizzare ogni parola nel seguente mondo :[C](VC)^m [V], dove *m* indica la lunghezza delle sottostringhe costituite da una vocale seguita da una consonante.

Esempio di parole con *m*=2:

Private, Oaten, Orrery;

Le regole per eliminare i suffissi implementate dall'algoritmo si presentano nella forma:

(CONDIZIONE) S1--> S2

Questo significa che se una parola che termina con il suffisso S1 e lo stem precedente soddisfa la condizione, allora S1 viene sostituito da S2.

Esempio

(m>1) EMENT -->
La condizione "m è maggiore 1", significa che deve esistere almeno una sequenza vocale-consonante all'interno dello stem della parola, se ciò si verifica, allora viene eliminato il suffisso "EMENT". Con tale regola, partendo dalla parola inglese "REPLACEMENT", si ottiene la parola "REPLAC".

Per comporre condizioni è possibile utilizzare gli operatori booleani (AND, OR, NOT).

Alcuni esempi di condizioni:

*S --> lo stem finisce con la consonante "S" oppure altra consonante. Per esempio: caresses, conditional, ;

v--> lo stem contiene una vocale. Per esempio: operator, airliner, irritant, activate;

*d--> lo stem finisce con una doppia consonante. Per esempio: decisiveness, hopefulness;

*o --> lo stem finisce con consonante-vocale-consonante e la seconda consonante è diversa "W", "X", "Y".

Per esempio: crepuscolar, consigned.

Ogni parola può essere suddivisa in due parti utilizzando il parametro *m* e leggendo la parola da sinistra verso destra:

quando la condizione *m* > 0 diventa vera: è la posizione dopo la prima consonante seguita da una vocale.
quando la condizione *m* > 1 diventa vera: è la posizione dopo la seconda coppia consonante-vocale dopo la prima consonante e la prima vocale.

Esempio

c r e p u s c u l a r

p1 p2
<---------R1------------>

<------R2------>

Questo metodo è utilizzato per calcolare la lunghezza di *m* e rimane invariato nei vari passi dell'algoritmo.

ALGORITMO (per lingua inglese).

Se la parola ha un numero di lettere <=2 rimane invariata.

Step 0) rimozione genitivi sassoni.

Step 1) elimina i suffissi ai verbi; ad esempio"-ing", "-ed" "-s".

Step 2 - 4) In questi tre passi vengono tolti i suffissi con la regola "*d" definita in precedenza.

Step 5) rimozione della vocale e dall'ultima posizione delle parole.

Step 6) rimozione delle doppie consonanti presenti al termine delle parole.

Nell'algoritmo è presente anche una soluzione al problema dell'over stemming, il quale viene risolto memorizzando in un dizionario, per ogni parola, la massima lunghezza del suffisso da eliminare.

Schema di pesatura

Il criterio adottato da Xapian (e richiamato da Recoll) per recuperare da una collezione i documenti ritenuti rilevanti in risposta ad una query si fonda su una variante della formula introdotta da Stephen Robertson nel 1976 che propose di pesare l'importanza dei termini presenti all'interno di una query nel seguente modo:

$$w(t) = log\frac{p(1-q)}{(1-p)q}$$

in cui:
t = termine presente nella query
p = probabilità che *t* reperisca un documento rilevante: p=P(t →d | d in $\mathcal{R}$)
q = probabilità che *t* reperisca un documento non rilevante: q=P(t→d | d non in $\mathcal{R}$)
(1-p) = probabilità che *t* non reperisca un documento rilevante: (1-p) = P(t NOT→ d | d in $\mathcal{R}$)
(1-q) = probabilità che *t* non reperisca un documento non rilevante: (1-q) = P(t NOT→ d | d non in $\mathcal{R}$)
d = documento
$\mathcal{R}$= insieme dei documenti rilevanti

Data una collezione, possiamo allora stimare p e q in questo modo:

$$p = \frac{r}{R}$$

$$(1-p) = \frac{R-r}{R}$$

$$q = \frac{n-r}{N-R}$$

$$(1-q) = \frac{N-R-n+r}{N-R}$$

indicando con

r = numero di documenti rilevanti reperiti da t
R= cardinalità di $\mathcal{R}$ (numero di tutti i documenti rilevanti)
N= numero di documenti che formano la collezione
n = numero di documenti reperiti dal termine t

I valori di probabilità così ottenuti, se sostituiti nella formula iniziale, forniscono una stima w(t):

$$w(t) = log\frac{r(N-R-n+r)}{(R-r)(n-r)}$$

Per evitare valori indefiniti della funzione logaritmo (per esempio quando n=r o r=0) è stata inoltre introdotta una costante h = ½ che trasforma la formula nella sua versione finale usata da Xapian:

$$w(t) = log\frac{(r+h)(N-R-n+r+h)}{(R-r+h)(n-r+h)}$$

in cui si ha che n dipende da t, R dalla query Q mentre r dipende si da t che da Q.

Se è vero che lo schema può apparire inefficace in quanto non si conosce a priori il valore di R, (ossia il reale numero di documenti rilevanti), è altrettanto vero che possiamo stimare p e q considerando un opportuno sottoinsieme di $\mathcal{R}$, calcolando poi i pesi dei termini contenuti nella query.

Lo schema di pesatura prevede inoltre l'assegnazione di un peso anche ai documenti, valore ottenuto sommando i w(t) dei termini di una query che fanno riferimento ad uno stesso documento d secondo la seguente formula:

$$w(d) = \sum_{t \to d, t \in Q} \frac{(r+h)(N-R-n+r+h)}{(R-r+h)(n-r+h)}$$

dove:
f_t = frequenza del termine t all'interno del documento D
L = lunghezza normalizzata del documento[1]
K= costante che dipende dalle caratteristiche della collezione considerata per la quale si consiglia un valore "basso" senza tuttavia specificare una regola di assegnazione.

Con questo criterio, il sistema reperirà un insieme definito Match Set costituito da K documenti in ordine decrescente di probabilità di rilevanza, richiamati da almeno un termine presente nella query e con valore w(d) > 0.

[1] la lunghezza normalizzata è data dal rapporto tra le m parole che costituiscono d e la lunghezza media dei documenti che costituiscono la collezione

Codice programmi sviluppati

```
/*Questo programma divide il file "time.txt" in tanti file quanti sono i testi presenti
al suo interno. Rinominando ciascun file come "num.txt" dove num e' il numero indicato
nel testo. */
import java.io.*;

public class SplitCollection
{
     public static void main (String[] args) throws IOException
      {
          BufferedReader in = new BufferedReader (new FileReader ("Time.txt"));
          PrintStream out = null;
          FileOutputStream fout = null;
          String str = null;
          int incr = 1;
          while ((str = in.readLine()) != null)
            {
              System.out.println ("Letto: " + str);
              if (str.indexOf ("*TEXT ") >= 0)
                 {
                   if (out != null && fout != null)
                      {
                           out.close ();
                       fout.close ();
                      }

                   fout = new FileOutputStream (incr + " .txt");
                   out = new PrintStream (fout);
                   System.out.println ("Apertura del documento " + incr);
                   incr++;
                 }
              else if (str.indexOf ("*STOP") > 0)
                   break;
              else
                   out.println (str);
          }
          out.close ();
          fout.close ();
          in.close ();
      }
}
```

```
// Questo programma costruisce un file da dare in input all'applicazione trec_eval par-
tendo
// da diversi file, ciascuno dei quali contiene i risultati di una query.

import java.io.*;
import java.util.*;

public class crearisultati
{
  public static void main (String[] args) throws IOException, FileNotFoundException
  {
     String in = args[0];    // viene passato da riga di comando il file iniziale
     String fi = args[1];     // viene passato da riga di comando il file finale
     int k= 999999;
     int inizio = Integer.parseInt(in);
     int fine = Integer.parseInt(fi);
     String file = "risultati.txt";
    // viene creato il file "risultati.txt"
    PrintWriter writer = new PrintWriter(new FileWriter(file));
/* nei cicli che seguuono, per ogni file contenente risultati di una query, viene letto
da ciascuna riga un il nome di un documento reperito e scrive nel file "risultati.txt"
una stringa con i parametri necessari a trec_eval, gestendo gli incrementi e i decrementi
necessari per le variabili numeriche. */

     while (inizio <= fine) {
          int r=1;
          BufferedReader reader = new BufferedReader(new FileReader("q"+inizio+".txt"));
          String id= reader.readLine();
          while (id!=null) {
               StringTokenizer st=new StringTokenizer(id);
               String num= st.nextToken();
               String s = inizio + " Q0 " + num + " "+ r + " "+ k + " STANDARD";
               writer.println(s);
               k--;
               r++;
               id=reader.readLine();
            }
      inizio++;
     }
     writer.close();
  }
 }
```

```
// Questo programma elimina le stop words dal testo "time.txt", creando un //file
"notime.txt". Come lista di stop words abbiamo utilizzato quelle
//indicate nella dispensa di Reperimento Dell'informazione.

import java.io.*;
import java.util.*;

public class Stopoutput
 {
  public static void main (String[] args) throws IOException, FileNotFoundException
  {
     BufferedReader reader = new BufferedReader(new FileReader(args[0]));
     File file = new File ("notext.txt");
      FileWriter filew = new FileWriter(file);
      BufferedWriter buf = new BufferedWriter(filew);

     PrintWriter print_writer = new PrintWriter (buf,true);
     int i = 0;

     String riga;
     while((riga = reader.readLine()) != null)
   {
      System.out.println(riga);
            String n= "";
           String s;
         boolean g=false;
         StringTokenizer st = new StringTokenizer(riga);
         while(st.hasMoreTokens())
                        {
              s=st.nextToken();
              BufferedReader buf1 = new BufferedReader(new FileReader("stoplistok.txt"));
              String test=buf1.readLine();
             g=false;
             while (test!=null)
                    {
                     if (s.equals(test))
                         { g=true; }
                     test=buf1.readLine();
                    }
               if(g==false)
                     { n = n + s + " "; }
         print_writer.println(n);
     }
  }
}
```

Risultati Trec_eval

Risultati interrogazioni Interrogazioni con operatore AND mantendo le stop word e mantenendo lo stemming

num_q	all	10
num_ret	all	17
num_rel	all	51
num_rel_ret	all	8
map	all	0.2956
gm_ap	all	0.0123
R-prec	all	0.3372
bpref	all	0.3372
recip_rank	all	0.6000
ircl_prn.0.00	all	0.6000
ircl_prn.0.10	all	0.4000
ircl_prn.0.20	all	0.4000
ircl_prn.0.30	all	0.4000
ircl_prn.0.40	all	0.3500
ircl_prn.0.50	all	0.2500
ircl_prn.0.60	all	0.2000
ircl_prn.0.70	all	0.2000
ircl_prn.0.80	all	0.2000
ircl_prn.0.90	all	0.2000
ircl_prn.1.00	all	0.2000
P5	all	0.1600
P10	all	0.0800
P15	all	0.0533
P20	all	0.0400
P30	all	0.0267
P100	all	0.0080
P200	all	0.0040
P500	all	0.0016
P1000	all	0.0008
recall5	all	0.3372
recall10	all	0.3372
recall15	all	0.3372
recall20	all	0.3372

recall30	all	0.3372
recall100	all	0.3372
recall200	all	0.3372
recall500	all	0.3372
recall1000	all	0.3372

Risultati interrogazioni mantenendo le stop word e mantenendo lo stemming con operatore OR

num_q	all	83
num_ret	all	34139
num_rel	all	324
num_rel_ret	all	323
map	all	0.6449
gm_ap	all	0.4782
R-prec	all	0.5875
bpref	all	0.9992
recip_rank	all	0.7404
ircl_prn.0.00	all	0.7610
ircl_prn.0.10	all	0.7586
ircl_prn.0.20	all	0.7379
ircl_prn.0.30	all	0.7300
ircl_prn.0.40	all	0.7148
ircl_prn.0.50	all	0.6965
ircl_prn.0.60	all	0.6223
ircl_prn.0.70	all	0.5867
ircl_prn.0.80	all	0.5648
ircl_prn.0.90	all	0.5356
ircl_prn.1.00	all	0.5270
P5	all	0.3880
P10	all	0.2819
P15	all	0.2137
P20	all	0.1693
P30	all	0.1161
P100	all	0.0381
P200	all	0.0194
P500	all	0.0078
P1000	all	0.0039
recall5	all	0.6224
recall10	all	0.7963
recall15	all	0.8575
recall20	all	0.8872
recall30	all	0.9024

recall100	all	0.9766
recall200	all	0.9984
recall500	all	0.9992
recall1000	all	0.9992

Risultati interrogazioni mantenendo le stop word e mantenendo lo stemming con operatore OR

num_q	all	21
num_ret	all	8763
num_rel	all	90
num_rel_ret	all	89
map	all	0.6365
gm_ap	all	0.5320
R-prec	all	0.5585
bpref	all	0.9968
recip_rank	all	0.7028
ircl_prn.0.00	all	0.7441
ircl_prn.0.10	all	0.7441
ircl_prn.0.20	all	0.7335
ircl_prn.0.30	all	0.7176
ircl_prn.0.40	all	0.7176
ircl_prn.0.50	all	0.7087
ircl_prn.0.60	all	0.6288
ircl_prn.0.70	all	0.5929
ircl_prn.0.80	all	0.5729
ircl_prn.0.90	all	0.5256
ircl_prn.1.00	all	0.5122
P5	all	0.4000
P10	all	0.3286
P15	all	0.2444
P20	all	0.1929
P30	all	0.1302
P100	all	0.0414
P200	all	0.0210
P500	all	0.0085
P1000	all	0.0042
recall5	all	0.6510
recall10	all	0.8537
recall15	all	0.9047
recall20	all	0.9179
recall30	all	0.9238
recall100	all	0.9905
recall200	all	0.9937

recall500	all	0.9968
recall1000	all	0.9968

Risultati interrogazioni mantenendo le stop word ma disabilitando lo stemming con operatore OR

num_q	all	21
num_ret	all	8763
num_rel	all	90
num_rel_ret	all	89
map	all	0.6365
gm_ap	all	0.5320
R-prec	all	0.5585
bpref	all	0.9968
recip_rank	all	0.7028
ircl_prn.0.00	all	0.7441
ircl_prn.0.10	all	0.7441
ircl_prn.0.20	all	0.7335
ircl_prn.0.30	all	0.7176
ircl_prn.0.40	all	0.7176
ircl_prn.0.50	all	0.7087
ircl_prn.0.60	all	0.6288
ircl_prn.0.70	all	0.5929
ircl_prn.0.80	all	0.5729
ircl_prn.0.90	all	0.5256
ircl_prn.1.00	all	0.5122
P5	all	0.4000
P10	all	0.3286
P15	all	0.2444
P20	all	0.1929
P30	all	0.1302
P100	all	0.0414
P200	all	0.0210
P500	all	0.0085
P1000	all	0.0042
recall5	all	0.6510
recall10	all	0.8537
recall15	all	0.9047
recall20	all	0.9179
recall30	all	0.9238
recall100	all	0.9905
recall200	all	0.9937
recall500	all	0.9968
recall1000	all	0.9968

Risultati interrogazioni eliminando le stop word ma mantenendo lo stemming con operatore OR

num_q	all	21
num_ret	all	3084
num_rel	all	90
num_rel_ret	all	88
map	all	0.6490
gm_ap	all	0.5308
R-prec	all	0.5923
bpref	all	0.9937
recip_rank	all	0.6834
ircl_prn.0.00	all	0.7300
ircl_prn.0.10	all	0.7300
ircl_prn.0.20	all	0.7300
ircl_prn.0.30	all	0.7300
ircl_prn.0.40	all	0.7300
ircl_prn.0.50	all	0.7234
ircl_prn.0.60	all	0.6407
ircl_prn.0.70	all	0.6165
ircl_prn.0.80	all	0.6025
ircl_prn.0.90	all	0.5463
ircl_prn.1.00	all	0.5399
P5	all	0.3905
P10	all	0.3238
P15	all	0.2381
P20	all	0.1929
P30	all	0.1302
P100	all	0.0419
P200	all	0.0210
P500	all	0.0084
P1000	all	0.0042
recall5	all	0.6431
recall10	all	0.8458
recall15	all	0.8813
recall20	all	0.9179
recall30	all	0.9238
recall100	all	0.9937
recall200	all	0.9937
recall500	all	0.9937
recall1000	all	0.9937

Risultati interrogazioni rimuovendo le stop word e disabilitando lo stemming con operatore OR

num_q	all	21
num_ret	all	3083
num_rel	all	90
num_rel_ret	all	88
map	all	0.6408
gm_ap	all	0.5257
R-prec	all	0.5718
bpref	all	0.9937
recip_rank	all	0.6834
ircl_prn.0.00	all	0.7227
ircl_prn.0.10	all	0.7227
ircl_prn.0.20	all	0.7227
ircl_prn.0.30	all	0.7227
ircl_prn.0.40	all	0.7179
ircl_prn.0.50	all	0.7114
ircl_prn.0.60	all	0.6346
ircl_prn.0.70	all	0.6052
ircl_prn.0.80	all	0.5912
ircl_prn.0.90	all	0.5344
ircl_prn.1.00	all	0.5281
P5	all	0.3810
P10	all	0.3143
P15	all	0.2413
P20	all	0.1929
P30	all	0.1302
P100	all	0.0419
P200	all	0.0210
P500	all	0.0084
P1000	all	0.0042
recall5	all	0.6374
recall10	all	0.8333
recall15	all	0.8893
recall20	all	0.9179
recall30	all	0.9238
recall100	all	0.9937
recall200	all	0.9937
recall500	all	0.9937
recall1000	all	0.9937

Bibliografia:

- Recoll: http://www.recoll.org
- Documentazione Recoll: http://www.lesbonscomptes.com/recoll/manuals.html
- Pacchetto installazione rpm recoll: http://www.lesbonscomptes.com/recoll/recoll-1.7.5-1.i386.rpm
- Pacchetti libiconv: http://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.11.tar.gz
- Motore xapian: http://www.xapian.org
- Pacchetto installazione xapian: http://www.oligarchy.co.uk/xapian/0.9.9/xapian-core-0.9.9.tar.gz
- Documentazione stemming: http://snowball.tartarus.org/algorithms/porter/stemmer.html
- Interfaccia QT: http://ftp.iasi.roedu.net/mirrors/ftp.trolltech.com/qt/source/qt-x11-opensource-src-4.2.2.tar.gz
- Contatto diretto tramite mail con lo sviluppatore Jean Francoins Dockes in data 5 Marzo 2007.