Sistemi Informativi
Laboratorio 3

Catalin Copil Mattia de Stefani Giulio Lovisotto

April 24, 2015

1 Descrizione

Abbiamo scelto di usare l'algoritmo di ranking BM25 . Tale algoritmo funziona nel modo seguente:

$$\sum_{i \in Q} \log\Big(\frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)}\Big) \cdot \frac{(k_1 + 1) f_i}{k + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

dove:

- i sono i termini della query Q,
- r_i e' il numero di documenti rilevanti che contiene il termine i,
- R e' il numero di documenti rilevanti per la query,
- n_i e' il numero di documenti che contiene il termine i nella collezione,
- N e' il numero totale di documenti nella collezione,
- k_1, k_2 sono parametri,
- f_i e' la frequenza del termine i nel documento,
- qf_i e' la frequenza del termine i nella query,

- K e' definito nel modo seguente:

$$K = k_1((1-b) + b \cdot \frac{dl}{avdl})$$

  dove b e' un parametro, dl e' la lunghezza del documento, $avdl$ e' la lunghezza media di un documento nella collezione. Questo termine serve a normalizzare il componente di frequenza rispetto alla lunghezza del documento (per non favorire i documenti troppo lunghi).

I termini R e r_i sono informazioni note a priori, tipicamente sono settati a zero in quanto non si hanno informazioni sulla rilevanza. Nel nostro caso li ignoriamo lasciandoli a zero.

2 Implementazione

Descriviamo ora le strutture dati necessarie al reperimento che vengono calcolate durante l'indicizzazione.

- matrice delle frequenze n_docs x n_words
- un array che contiene le lunghezze dei documenti

La funzione di reperimento scorrera' la lista di documenti (le righe della matrice), e per ogni documento scorrera' sui termini della query Q (*Document-at-a-time retrieval*). Poi i documenti con punteggio maggiore di zero verranno ordinati (dal maggiore al minore). Il seguente pseudocodice mostra la procedura di retrieval:

$Q \leftarrow query$
for all $doc \in C$ **do**
    $score \leftarrow 0$
    for all $qw \in Q$ **do**
        $score \leftarrow score + f_{bm25}(doc, qw)$
    end for
end for
return top k scoring documents

La complessita' di tale algoritmo e' $O(n \cdot m)$ dove n e' il numero di documenti nella collezione C, e m e' il numero di descrittori nell'interrogazione Q. Per i parametri abbiamo scelto: $k_1 = 0.01$, $k_2 = 1$, $b = 0$, in quanto fornivano la migliore precisione nei risultati. Il termine b a zero significa che non viene applicato alcuno *smoothing* sulle frequenze rispetto alla lunghezza del documento, nel nostro caso cio' funziona

perche' i documenti sono in genere molto corti. Il termine k_1 e' stato settato ad un valore molto basso in quanto essendo i documenti molto corti (per alcuni ci sono solo 2-3 parole), la componente di frequenza nella formula di BM25 aveva un peso troppo influente rispetto alle altre.

3 Risultati

Abbiamo valutato i risultati della nostra implementazione con l'utility *trec_eval* e abbiamo ottenuto i risultati riportati in figura.

```
runid                   all     G12R9
num_q                   all     43
num_ret                 all     4300
num_rel                 all     719
num_rel_ret             all     406
map                     all     0.3156
```

Figure 1: Risultati *trec_eval*

Si puo' vedere che la *map* e' piuttosto bassa. Indagando sulle query abbiamo notato che alcune di esse contengono bisogni informativi legati agli autori ("I am interested in articles written either by Prieve or Udo PoochPrieve, B. Pooch, U."), e nella nostra implementazione cio' non e' colto in quanto nei documenti indicizzati il campo autori e' ignorato (nei file non sono presenti gli stem degli autori). Inoltre non sempre gli stem presenti nei file forniti rappresentano sempre il bisogno informativo ("Interested in articles on robotics, motion planning particularly the geometric and combinatorial aspects. We are not interested in the dynamics of arm motion.") in quanto il modello scelto non permette di esprimere condizioni booleane.

3.1 Lucene

Per confrontare i risultati abbiamo utilizzato il modulo PyLucene (python api per lucene). Abbiamo indicizzato i seguenti campi dei documenti: autore, abstract, titolo, prendendoli dal testo originale in xml. Figura 2 riporta i risultati ottenuti.

Come si puo' vedere la *map* e' leggermente inferiore nonostante l'utilizzo del campo autori, che nella nostra implementazione non e' presente. Cio' e' dovuto alla

```
runid                   all     G12R0
num_q                   all     43
num_ret                 all     40039
num_rel                 all     719
num_rel_ret             all     570
map                     all     0.2972
```

Figure 2: Risultati *trec_eval* sull'output di lucene

funzione di ranking di lucene, che combina modello vettoriale e probabilistico ma non tiene conto della brevita' dei documenti della nostra collezione.