Documento finale per la prova d'esame di Sistemi Informativi a.a. 2014/2015

Catalin Copil
catalin.copil@studenti.unipd.it
Mattia de Stefani
mattia.destefani@studenti.unipd.it
Giulio Lovisotto
giulio.lovisotto@studenti.unipd.it

Sommario Il presente documento descrive le attività di laboratorio svolte dagli autori (gruppo 12) durante il corso di Sistemi Informativi, tenuto dal professor M. Melucci presso l'Università di Padova, anno accademico 2014/2015. L'obiettivo delle tecniche utilizzate è quello di massimizzare l'efficacia della funzione di reperimento nella collezione di articoli scientifici CACM. Questo documento riporta informazioni sulle scelte implementative dei metodi di Information Retrieval trattati. I componenti del gruppo hanno deciso di integrare nel reperimento le tecniche proposte partendo da BM25 come riferimento. In particolare, sono stati integrati gli algoritmi di PAGERANK, RELEVANCE FEEDBACK, LSA, HITS. Inoltre ai fini dell'ottimizzazione è stato utilizzata un Evolution Strategy. Il documento comprende un'analisi dei risultati ottenuti, e termina con alcune conclusioni generali sul lavoro svolto.

1 Introduzione

Il corso di Sistemi Informativi ha come obiettivo l'insegnamento di nozioni basilari di Information Retrieval e Motori di Ricerca. Gli argomenti trattati a lezione ed i concetti applicati durante i laboratori sono stati utili a comprendere il funzionamento di un sistema di Information Retrieval.

I progetti svolti nelle lezioni di laboratorio prevedono l'utilizzo della CACM collection, che è una collezione di articoli scientifici tratti dalla rivista *Communications of the ACM* [2]. La collezione contiene i titoli, gli abstract, e altre informazioni quali: autori, anno di pubblicazione, la lista di citazioni verso altri documenti della collezione. Inoltre è disponibile un set di query e relativi giudizi di rilevanza. In questa collezione la ricerca si è concentrata sull'utilizzo dei soli titoli e abstract per il reperimento, quindi verrà utilizzato tale approccio.

Grazie all'insieme delle citazioni è possibile combinare la funzione di reperimento con l'uso di Pagerank e Hyperlink Induced Topic Search (HITS). Tali algoritmi utilizzano il grafo costruito con le citazioni, dove i nodi sono documenti, ed è presente un arco dal nodo u al nodo v se il documento u contiene nelle sue *references* un riferimento al documento v. A partire dal grafo calcolano dei punteggi per ogni nodo, ed essi vengono integrati nella funzione di reperimento al fine di ottenere una maggiore efficacia. È stata implementata una funzione di reperimento che facesse uso di Latent Semantic Analysis (LSA), che è una tecnica che sfrutta una riduzione di dimensionalità per individuare i concetti che meglio esprimono la rilevanza per una query. Inoltre è stato utilizzato il Relevance Feedback (RF), che si avvale delle informazioni di rilevanza (note a priori o per assunzione), per riordinare i documenti reperiti secondo il modello di rilevanza individuato. Infine è stato implementato un algoritmo di Evolution Strategy (ES) per l'ottimizzazione degli algoritmi di reperimento trattati. L'esposizione del presente documento si svolge secondo criteri di

chiarezza e sintesi. Per tutti i dettagli che non vengono descritti si rimanda agli autori e ai testi di riferimento [5,6,4].

1.1 Organizzazione

Il resto del documento è organizzato come segue. In Sezione 2 vengono descritti i metodi sviluppati per ciascun laboratorio. In Sezione 3 vengono discussi i risultati ottenuti per i vari laboratori, e vengono descritte alcune considerazioni sull'efficienza dei metodi utilizzati. Infine Sezione 4 chiude il documento con delle conclusioni generali.

2 Metodologia

In questa sezione verranno descritti in dettaglio l'approccio usato per il lavoro di gruppo e le funzioni di reperimento realizzate.

2.1 Approccio

L'obiettivo delle funzioni di reperimento implementate é di massimizzare la Mean Average Precision (MAP). Per calcolarla e verificarne il cambiamento durante le varie versioni del codice viene utilizzato `trec_eval`, esso è il tool standard usato dalla TREC community per valutare l'efficacia del reperimento. Tale strumento prende in input un file di risultati del reperimento e un file con i giudizi di rilevanza *qrels-treceval.txt*, e fornisce un insieme di misure di efficacia.

È stato scelto di utilizzare un metodo probabilistico per la risoluzione dei laboratori, in particolare è stato utilizzato l'algoritmo BM25 [6] poiché é un buon modello, ed è stato usato con successo in letteratura [4]. Per realizzare le implementazioni è stato utilizzato Python, corredato da diverse librerie: `numpy/scipy` per la gestione delle matrici, `matplotlib` per il plot dei risultati, `networkx` per la gestione dei grafi. Per il versionamento si è scelto `Git`. Tutto il codice prodotto è disponibile online sulla piattaforma GitHub [1].

Per lo svolgimento dei laboratori è stato utilizzato un approccio di gruppo. Durante le ore di laboratorio i componenti del gruppo affrontavano una discussione, riguardante sia gli aspetti teorici che pratici trattati. Veniva quindi deciso come procedere, e durante il successivo incontro venivano rifiniti i dettagli del lavoro svolto e venivano prodotti i documenti da consegnare.

2.2 Indicizzazione

Descrizione. Costruzione di un algoritmo di indicizzazione in grado di costruire delle strutture dati necessarie al reperimento con BM25.

Implementazione. Per l'implementazione è stato utilizzato il file *freq.docid.stem.txt* che contiene le frequenze delle keywords (dopo lo stemming) nei documenti. Poiché le stop words sono già state rimosse da tale file, viene ignorata la stoplist. L'algoritmo di indicizzazione utilizza le keywords presenti in tale file per costruire un dizionario delle keywords presenti nella collezione. In seguito scorre sulle righe del file *freq.docid.stem.txt* e costruisce la Document Term Matrix (DTM), una matrice di dimensione $n_{docs} \times n_{words}$ che contiene in posizione (j, i) il numero di occorrenze del termine j nel documento i. Infine salva su file le seguenti strutture:

– Le lunghezze (numero di termini) dei documenti, necessarie alla normalizzazione sulla lunghezza in BM25,
– La matrice DTM,
– La DTM normalizzata sulle colonne, cioè una matrice delle frequenze che contiene in posizione (j, i) la frequenza del termine i nel documento j, nel resto del documento verrà fatto riferimento a questa matrice come FDTM,
– Il dizionario contenente le keywords della collezione.

2.3 Reperimento

Descrizione. Comprensione dei meccanismi di reperimento e costruzione di un primo algoritmo di reperimento. Realizzazione dell'algoritmo BM25, che riceve in input una query e restituisce una lista ordinata di documenti con i relativi punteggi. Per chiarezza viene riportata la formula del reperimento tramite tale algoritmo (per il documento j), si rimanda al testo di riferimento [6] per la spiegazione completa:

$$S_j^{(BM25)} = \sum_{i \in Q} \log\Big(\frac{N - n_i + 0.5}{n_i + 0.5}\Big) \cdot \frac{(k_1 + 1) f_i}{k + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}.$$

Implementazione. L'algoritmo scorre la lista dei documenti, e per ognuno di essi calcola e somma il punteggio di BM25 per ogni termine della query $qw \in Q$. Le strutture ottenute dall'indicizzazione (Sezione 2.2) vengono utilizzate per tale calcolo. Il dizionario e la matrice FDTM vengono utilizzati per recuperare le frequenze dei termini, mentre le lunghezze dei documenti vengono direttamente inserite nella formula di BM25. Lo pseudocodice per l'algoritmo è il seguente:

$C \leftarrow documents$
$Q \leftarrow query$
$S \leftarrow list()$
for all $doc \in C$ **do**
  for all $qw \in Q$ **do**
    $S_{doc}^{(BM25)} = S_{doc}^{(BM25)} + BM25(qw)$
  end for
end for
$sort(S)$
return top k scoring documents

Tale algoritmo ha complessità $O(n \cdot m)$, dove n è il numero di documenti nella collezione ed m è il numero di termini presenti nella query Q. Il risultato della funzione di reperimento qui presentata verrà indicato nel resto del documento come BASELINE. Ai fini dell'efficienza, i due cicli dell'algoritmo sono stati in un secondo momento convertiti in una serie di operazioni su matrici. Grazie ai benefici della *vectorization* in `numpy`, ciò ha garantito un notevole miglioramento del tempo di esecuzione.

2.4 Relevance Feedback

Descrizione. Implementazione di una funzione di reperimento che utilizza Relevance Feedback, sia ESPLICITO che PSEUDO. Nel primo caso, si ricorda che l'algoritmo BM25 nella sua forma completa,

integra le informazioni di rilevanza:

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

dove R è il numero di documenti rilevanti per la query considerata, mentre r_i è il numero di documenti rilevanti che contengono il termine i. Mentre nel caso del RF ESPLICITO tali informazioni sono note, nel caso di RF PSEUDO esse vengono ricavate dopo un primo reperimento con la funzione BASELINE, dove i primi N documenti sono considerati rilevanti.

Implementazione. L'implementazione delle due tipologie di RF è differente, poiché il calcolo dello PSEUDO richiede due passi, mentre ne basta uno solo per l'ESPLICITO.

- ESPLICITO: viene utilizzato il file *qrels-treceval.txt*, contenente i giudizi di rilevanza, per estrarre i valori di R ed r_i.
- PSEUDO: viene eseguito un reperimento con la funzione BASELINE. Vengono poi considerati rilevanti i primi N documenti reperiti, cioè $R = N$. Per il calcolo di r_i viene utilizzata FDTM, dove un elemento $FDTM(j, k) > 0$ indica che il documento j contiene il termine k.

I valori di R, r_i vengono inseriti nella formula di BM25 per ottenere il ranking finale.

2.5 PageRank

Descrizione. Comprensione di PAGERANK e implementazione di una funzione di reperimento che lo integri. È stata scelta la seguente formula per il ranking. Per ogni documento j, essa combina gli score di BM25 ($S_j^{(BM25)}$) con il valore del PAGERANK (pr_j) nel seguente modo:

$$S_j^{(PR)} = \alpha \cdot S_j^{(BM25)} + (1 - \alpha) \cdot pr_j,$$

dove α è un parametro che varia tra 0 ed 1 e, quindi, sposta il peso uniformemente da PAGERANK a BM25. Per poter effettuare tale somma in modo coerente, entrambe le distribuzioni (gli score di BM25 e i valori di PAGERANK) devono essere ridimensionate. È stato scelto di normalizzare gli $S_j^{(BM25)}$ e i pr_j in modo da avere valori compresi tra 0 e 1. Per far ció abbiamo usato la formula:

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)},$$

dove z_i é il valore ridimensionato, e x_i é il valore originale.

Implementazione. Per effettuare il calcolo del PAGERANK abbiamo utilizzato la libreria `networkx`, essa attraverso il metodo `read_edgelist` permette la costruzione del grafo delle citazioni a partire da una lista di archi, contenuta nel file *citazioni.txt*. Inoltre dispone del metodo `pagerank` per la computazione del PAGERANK utilizzando POWER ITERATION sulla matrice di transizione. Tale computazione è fatta a tempo di indicizzazione, e i valori sono salvati su un file. A tempo di reperimento l'algoritmo effettua il reperimento con la funzione BASELINE e poi combina i risultati usando i valori di PAGERANK per calcolare il punteggio finale. La normalizzazione viene effettuata con metodi e operazioni su vettori.

2.6 Latent Semantic Analysis

Descrizione. Comprensione di Latent Semantic Analysis e costruzione di una funzione di reperimento che integri LSA. È stato scelto di utilizzare LSA per riordinare i primi N documenti reperiti con la funzione di reperimento BASELINE.

Implementazione. L'algoritmo effettua il reperimento con la funzione BASELINE, e prende in considerazione soltanto i primi N documenti reperiti. Filtra quindi la matrice DTM in maniera che le uniche righe considerate siano quelle relative agli N documenti considerati, e successivamente rimuove i termini (le colonne) che non compaiono in tale matrice. Ci riferiamo alla matrice così ottenuta come X, che rappresenta ora il nostro spazio delle feature. Viene poi utilizzato il metodo `numpy.linalg.svd` per il calcolo della fattorizzazione della matrice X:

$$X = U\Sigma V^T.$$

Vengono poi considerate m dimensioni, ottenendo quindi la matrice ridotta:

$$X_m = U_m \Sigma_m V_m^T.$$

Viene poi effettuata la proiezione del vettore query $\boldsymbol{q}$ sullo spazio a dimensione ridotta secondo la formula:

$$\boldsymbol{q}_m = \Sigma_m^{-1} U_m^T \boldsymbol{q}.$$

Infine l'algoritmo calcola la *cosine similarity* tra le rappresentazioni degli N documenti nello spazio ridotto (cioè le colonne V_m^T) e la query ridotta $\boldsymbol{q}_m$. Il ranking per gli N documenti viene modificato rispetto alla similarità decrescente cosi calcolata. I restanti documenti vengono semplicemente accodati.

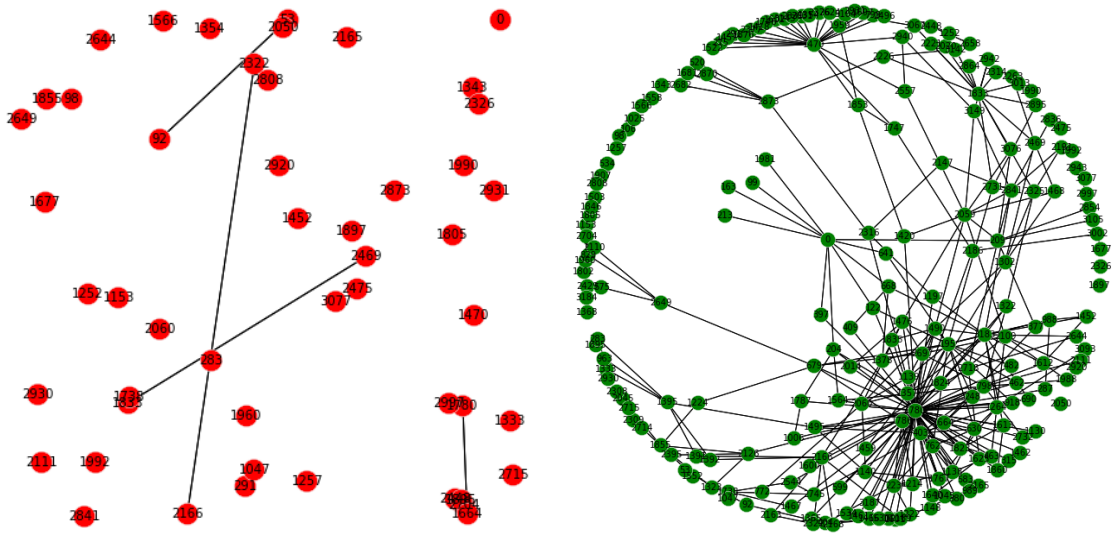
2.7 Hyper-linked Induced Topic Search

Descrizione. Comprensione di Hyperlinked Induced Topic Search, implementazione di una funzione di reperimento che integri HITS. È stato scelto di riordinare i primi N documenti reperiti con la funzione di reperimento BASELINE. A tal fine viene costruito il grafo radice G_R contente gli N documenti individuati, ed esso viene espanso secondo la dinamica standard: ogni nodo citato dai nodi di G_R o citante i nodi di G_R viene aggiunto al grafo radice di partenza. Il grafo risultante è detto grafo base G_B. Infine per ogni documento j di G_B, viene calcolato lo score $S_j^{(HITS)}$ con la seguente formula:

$$S_j^{(HITS)} = \alpha \cdot S_j^{(BM25)} + \beta \cdot auth_j + \gamma \cdot hub_j,$$

dove $auth_j$ e hub_j indicano rispettivamente il punteggio di autorevolezza e di hubbines per il documento j nel grafo G_B. Il valore dei parametri α, β, γ può variare tra [0, 1]. Infine gli N documenti inizialmente individuati vengono riordinati in base al punteggio così ottenuto. Una normalizzazione come quella presentata in Sezione 2.5 viene utilizzata per ottenere dei punteggi confrontabili per le tre distribuzioni (autorevolezza, hubbiness, punteggio di BM25).

Implementazione. L'algoritmo effettua il reperimento con la funzione BASELINE, e prende in considerazione soltanto i primi N documenti reperiti. Carica poi l'intero grafo delle citazioni, utilizzando la lista di archi contenuta in *citazioni.txt*. Poi viene salvata la lista dei nodi che costituisce il grafo radice G_R, e viene effettuata l'espansione iterando sugli archi di tali nodi, ottenendo così un'altra lista di nodi che formano G_B, il grafo base. Su quest'ultimo grafo viene computato HITS con il metodo `networkx.hits`. Le distribuzioni dei punteggi vengono ridimensionate, e infine viene calcolato $S_j^{(HITS)}$. Infine l'algoritmo modifica il ranking per gli N documenti rispetto al punteggio calcolato. I restanti documenti vengono semplicemente accodati. Figura 1 mostra il grafo radice e il grafo base per la Query 56.



(a) Grafo radice $G_R^{(56)}$

(b) Grafo base $G_B^{(56)}$.

Figura 1: Grafi costruiti a partire dai primi $N = 50$ documenti reperiti per la Query 56.

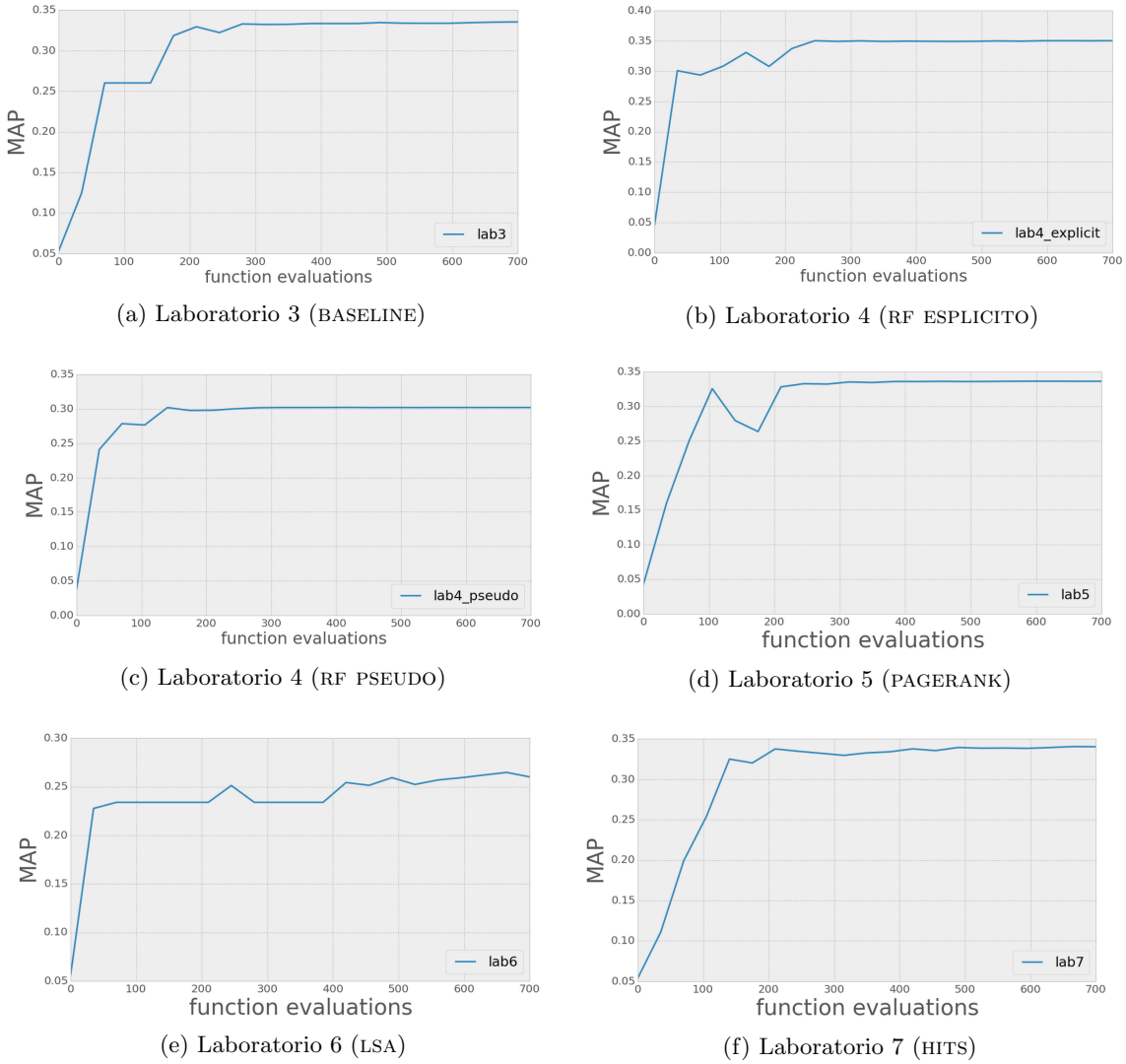
2.8 Evolution Strategy

Per ottimizzare gli algoritmi di reperimento è stato scelto di utilizzare un Evolution Strategy [3], una tecnica di ottimizzazione basata sui principi che regolano l'evoluzione. Tecniche di questo tipo sono più robuste rispetto ai metodi di ricerca lineare per quanto riguarda i massimi locali. Il loro svantaggio consiste nel maggior numero di valutazioni richieste. In questo contesto una valutazione impiega circa 1-17 secondi a seconda della complessità del metodo di reperimento. Ciò permette di eseguire l'algoritmo di ottimizzazione in un tempo accettabile.

I parametri scelti per l'ottimizzazione cambiano in base alla funzione di reperimento (e quindi del laboratorio). Per il laboratorio 3 (BASELINE), 4 (RF) e 6 (LSA) sono stati ottimizzati k_1 e b. È stato ignorato k_2 in quanto non ci sono termini ripetuti nelle query e quindi quest'ultimo non influisce sul punteggio. Per quanto riguarda il laboratorio 5 (PAGERANK) vengono ottimizzati k_1, b ed α. Invece per il laboratorio 7 (HITS) vengono ottimizzati k_1, b, α, β e γ. Per lo pseudo relevance

feedback del laboratorio 4 il numero di documenti considerati rilevanti $R = 50$, mentre per l'LSA del laboratorio 6 il numero di documenti da riordinare $N = 30$ e il numero di dimensioni per la riduzione $m = 2$ (tali parametri sono interi e non si prestano ad un'ottimizzazione tramite ES). Per HITS laboratorio 7 il numero di elementi nel grafo radice $N = 60$. La funzione da massimizzare è la Mean Average Precision.

Figura 2 riporta l'andamento della MAP durante l'ottimizzazione della funzione di reperimento dei laboratori.



(a) Laboratorio 3 (BASELINE)

(b) Laboratorio 4 (RF ESPLICITO)

(c) Laboratorio 4 (RF PSEUDO)

(d) Laboratorio 5 (PAGERANK)

(e) Laboratorio 6 (LSA)

(f) Laboratorio 7 (HITS)

Figura 2: Valore MAP durante iterazioni dell'ES, per le funzioni di reperimento dei vari laboratori.

Tabella 1 riporta i risultati dell'ottimizzazione.

metodo	parametri ottimizzati	MAP
BASELINE	$k_1 = 0.0253, b = 0.0221$	0.3354
RF ESPLICITO	$k_1 = 0.02065, b = 0.0$	0.3504
RF PSEUDO	$k_1 = 0.02065, b = 0.0$	0.3022
PAGERANK	$k_1 = 0.0237, b = 0.0, \alpha = 0.7832$	0.3366
LSA	$k_1 = 0.0087, b = 0.0$	0.2648
HITS	$k_1 = 0.0247, b = 0.0185, \alpha = 1.0, \beta = 0.1098, \gamma = 0.0549$	0.3414

Tabella 1: Risultati ottimizzazione con ES per le funzioni di reperimento dei vari laboratori.

Grazie all'ES si è ottenuto un notevole miglioramento rispetto alla prima versione dove i parametri sono stati scelti manualmente e il valore di MAP si aggirava attorno al 0.25. Inoltre dai grafici si può notare come l'algoritmo di ottimizzazione sia molto robusto, dato che dopo ~ 300 valutazioni è già molto vicino alla configurazione ottima. I risultati dell'ottimizzazione vengono discussi in Sezione 3.

3 Risultati sperimentali

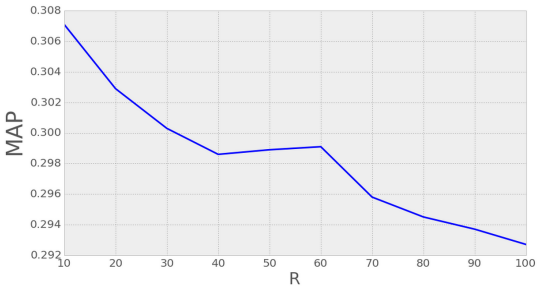
In questa sezione verranno discussi i risultati sperimentali dei vari laboratori. Per confrontare le diverse funzioni di reperimento la configurazione di parametri per BM25 è fissata a quella riportata in tabella 1, cioè $k_1 = 0.0253, b = 0.0221$, per ogni diverso laboratorio. Il risultato del reperimento con il laboratorio 3 che utilizza tale configurazione è la BASELINE. È interessante notare che con l'ottimizzazione i valori trovati per k_1, b sono molto lontani dai valori noti in letteratura ($b = 0.75, k_1 = 1.2$). Tali valori rendono il secondo termine della funzione di reperimento di BM25,

$$\frac{(k_1 + 1) f_i}{k_1(1 - b + b \cdot \frac{dl}{avdl}) + f_i},$$

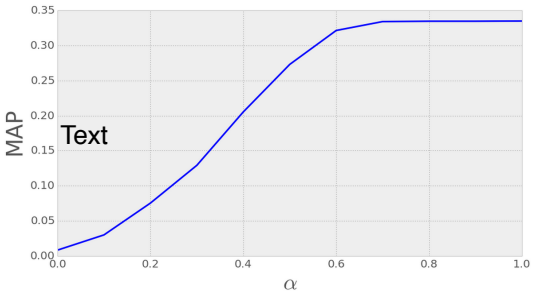
tendente a 1. Ciò significa che nel nostro contesto, la normalizzazione basata sulle lunghezze del documento non è efficace, e inoltre che frequenze molto diverse di occorrenza dei termini portano a contributi nella funzione di reperimento molto simili. Tale fenomeno può essere giustificato con il fatto che la maggior parte dei nostri documenti contiene pochi termini (solo i titoli), quindi la frequenza dei termini è molto meno importante rispetto alla loro presenza. Infatti anche per frequenze molto basse il contributo di questo termine è vicino a 1.

Figura 3 mostra l'andamento della MAP per le varie funzioni di reperimento, al variare di alcuni parametri della funzione di reperimento.

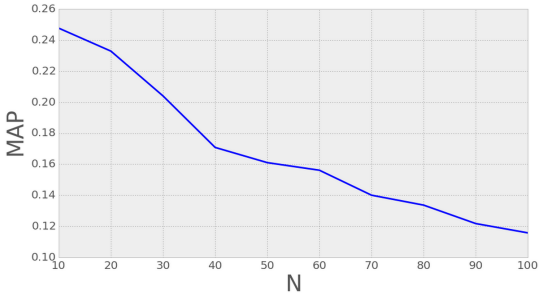
Come si può vedere da Figura 3a e Figura 3c, il RF PSEUDO e l'LSA peggiorano il valore della MAP al crescere del numero di documenti considerati rilevanti, e del numero di documenti da riordinare, rispettivamente. Ciò significa che dopo il primo passaggio di BM25 l'ordine dei documenti è già preciso, e che l'effetto dello pseudo RF e di LSA è quello di alterare tale ordine, peggiorando l'efficacia del reperimento. In Figura 3b vediamo che α il parametro che regola l'influenza di PAGERANK sull'ordine dei documenti, ha un effetto molto negativo per valori piccoli, mentre migliora per valori



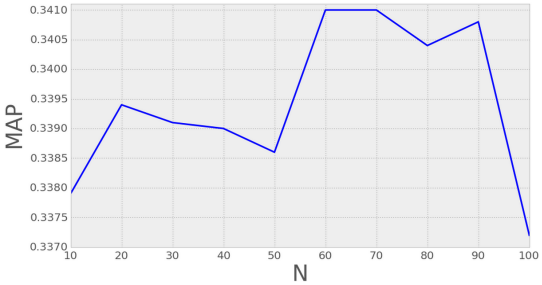
(a) Laboratorio 4 (RF PSEUDO), al variare di R.



(b) Laboratorio 5 (PAGERANK), al variare di α.



(c) Laboratorio 6 (LSA), al variare di N.



(d) Laboratorio 7 (HITS), al variare di N.

Figura 3: MAP al variare di alcuni parametri di configurazione della funzione di reperimento, per i diversi laboratori.

più alti. Infine in Figura 3d vediamo che il reperimento che si avvale di HITS ha un massimo per $N = 60$, il che significa che se andiamo ad espandere un insieme radice di 60 elementi otteniamo in media la miglior MAP dopo il riordinamento.

Figura 4 mostra le MAP dei vari metodi di reperimento a partire dalla configurazione BASELINE di BM25 (laboratorio 3), scegliendo la miglior combinazione di parametri per ciascuno di essi. Tabella 2 riporta i parametri usati per ogni funzione di reperimento e la relativa MAP ottenuta.

Dai risultati riportati si può notare che i metodi che hanno portato un miglioramento in termini di MAP rispetto alla BASELINE, sono il RF ESPLICITO, e HITS. Il notevole peggioramento portato da LSA può essere spiegato dalla rappresentazione dei documenti nello spazio vettoriale, infatti essi contentendo molto pochi termini in media, sono vettori con molti elementi a zero. Ciò significa che nel nostro contesto, dopo la riduzione dimensionale, le similarità tra i vettori non sono significative. In particolare la capacità di LSA di mitigare il problema di sinonimia non è efficace, probabilmente perchè la collezione è piuttosto ristretta. Il riordinamento quindi sposta alcuni documenti rilevanti dalla cima della classifica verso il fondo. Per il PAGERANK invece possiamo vedere che sebbene esso non abbia portato un miglioramento partendo dai parametri della BASELINE (per $\alpha = 1$ il contributo di PAGERANK viene completamente ignorato nel reperimento), utilizzando α come parametro da ottimizzare, tale metodo può portare un leggero miglioramento della precisione, come riportato in Tabella 1. Osservando che anche HITS ha portato un aumento della MAP in questi termini, possiamo affermare che l'analisi del grafo delle citazioni porta informazione utile alla rilevanza.

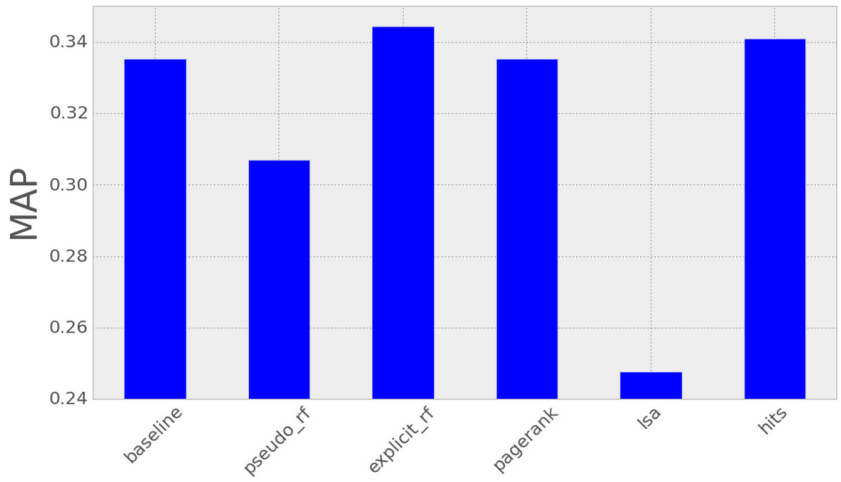


Figura 4: MAP per i vari metodi di reperimento, a partire dai parametri di BM25 della BASELINE.

metodo	parametri	MAP
BASELINE	-	0.3354
RF ESPLICITO	-	0.3445
RF PSEUDO	$R = 10$	0.3071
PAGERANK	$\alpha = 1.0$	0.3354
LSA	$N = 10, m = 2$	0.2478
HITS	$\alpha = 1.0, \beta = 0.1, \gamma = 0.1, N = 60$	0.3412

Tabella 2: Dettaglio dei parametri usati per le varie funzioni di reperimento e della relativa map. I parametri di BM25 per ogni metodo sono quelli della BASELINE ottimizzata $k_1 = 0.0253, b = 0.0221$.

3.1 Efficienza

Figura 5 riporta i tempi di esecuzione delle varie funzioni di reperimento, per l'intero set di query fornite. I parametri utilizzati sono quelli riportati in Tabella 2. Per ogni metodo si presume che l'indicizzazione sia già avvenuta, e inoltre tutte le misure calcolabili offline (a tempo di indicizzazione) siano già state computate (e.g.: il dizionario, le lunghezze dei documenti, la DTM, il valore di pagerank). Come si può vedere, in particolare grazie alla *vectorization* usata per effettuare il calcolo di BM25, il tempo di esecuzione del reperimento BASELINE è molto breve, poco più di un secondo. Gli altri metodi di reperimento sono invece più impegnativi. In particolare, i metodi LSA e HITS sono molto lenti, nel primo caso per la costosa decomposizione SVD e il calcolo delle cosine similarities, nel secondo per l'espansione del grafo radice. Gli altri metodi hanno comunque dell'overhead dovuto ai tempi di lettura dei dati addizionali e al riordinamento della lista di documenti.

Tali risultati sono da considerarsi indicativi, in quanto non sono state usate regole vincolanti sul codice volte a garantire l'efficienza. Inoltre la dimensione della collezione utilizzata ci ha permesso di effettuare maggior parte delle computazioni utilizzando matrici. Tale vantaggio non è applicabile al crescere del corpo di documenti.

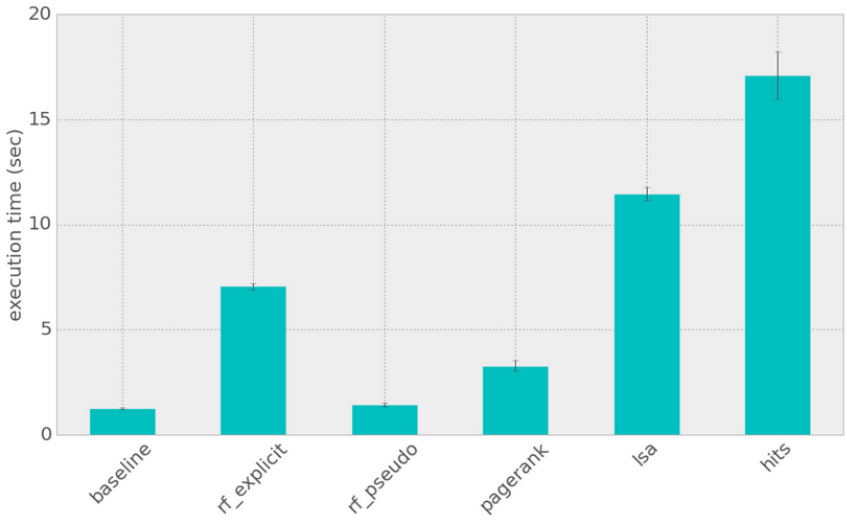


Figura 5: Tempo di esecuzione (secondi) e relativa standard deviation per le varie funzioni di reperimento sull'intero set di query. Media su 20 run. Le run vengono eseguite su thread singolo.

4 Conclusioni

Nell'arco del corso di Sistemi Informativi sono state affrontate diverse tecniche per il reperimento. Durante le implementazioni abbiamo notato che non sempre metodi più avanzati portavano un miglioramento rispetto alla BASELINE e anche quando questo succedeva, la configurazione dei parametri era molto suscettibile e delicata. Inoltre la MAP ottenuta è relativamente bassa rispetto alle nostre aspettative iniziali. Siamo quindi arrivati a domandarci i motivi di questo comportamento. Analizzando il testo delle query abbiamo avuto modo di verificare come alcune di esse hanno bisogni informativi che riguardano informazioni che non sono presenti nel corpo di documenti indicizzati. Molte di esse chiedono infatti informazioni quali l'autore o l'anno di pubblicazione, altre invece esprimono regole di esclusione (eg. Query 6: "[...] We are not interested in the dynamics of arm motion.") che non vengono considerate come tali dal modello che abbiamo utilizzato. Un altro aspetto che ha influenzato i risultati è il fatto che per molti documenti mancano gli abstract, e viene utilizzato solo il titolo. È difficile esprimere il vero contenuto informativo di un articolo utilizzando solamente le 2-10 parole del titolo. Riteniamo che questi fattori portino ad uno scarso miglioramento dell'efficacia del reperimento utilizzando tecniche avanzate, che potrebbero invece contribuire in maniera significativa su collezioni più grandi e complete.

Osservando le query abbiamo notato che alcune di esse sono formulate in modo colloquiale. Queste situazioni portano ad avere degli stem che non esprimono il bisogno informativo dell'utente. Un esempio è la Query 64: "List all articles on EL1 and ECL (EL1 may be given as EL/1; I don't remember how they did it.", trasformata negli stem *[articl, ecl, el, list, rememb]*. In questo caso lo stem *rememb* non è legato al bisogno informativo in quanto appartiene ad un commento personale dell'utente, nonostante ciò viene utilizzato per il reperimento. A tal proposito, si potrebbero sperimentare tecniche di preprocessing delle query, con l'obiettivo di raffinare la precisione del contenuto

e minimizzare il *topic drift*.

Abbiamo avuto modo di verificare il notevole miglioramento della MAP utilizzando RF ESPLICITO, che non risente pesantemente delle considerazioni qui fatte, in quanto modifica l'ordinamento basandosi solamente su informazioni di rilevanza e occorrenze dei termini, e non utilizza aspetti più complessi (e.g.: il grafo delle citazioni).

Un altro aspetto è stato gestire l'efficienza del reperimento. Sarebbe interessante verificare in che modo le tecniche realizzate si comportano con l'utilizzo di collezioni più grandi, e quali delle scelte effettuate per le implementazioni sulla collezione CACM si rivelano essere dei colli di bottiglia al crescere del corpo di documenti.

Riferimenti bibliografici

1. Github link. `https://github.com/giuliolovisotto/information-retrieval`.
2. CACM collection. `http://ir.dcs.gla.ac.uk/resources/test_collections/cacm/`.
3. T. Back. *Evolutionary algorithms in theory and practice*. Oxford Univ. Press, 1996.
4. W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010.
5. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge university press Cambridge, 2008.
6. M. Melucci. *Information retrieval. Metodi e modelli per i motori di ricerca*. Franco Angeli, 2013.