

UNIVERSITATEA BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INTELIGENȚĂ ARTIFICIALĂ

LUCRARE DE DISERTAȚIE

Metode de ierarhizare în Regăsirea Informației

Autor:
Călin AVASÎLCĂI

Coordonator:
Lect. Dr. Marius POPESCU

IUNIE 2010

Cuprins

1	Introducere	5
1.1	Scurt istoric	5
1.2	Descrierea procesului	5
1.3	Un exemplu	6
2	Regăsirea informației - generalități	9
2.1	Procesul de indexare	9
2.1.1	Colectarea textului	9
2.1.2	Transformarea textului	10
2.1.3	Crearea indexului	11
2.2	Procesul de căutare	11
2.2.1	Interacțiunea cu utilizatorul	11
2.2.2	Procesul de regăsire și ierarhizare	12
2.2.3	Evaluarea	14
3	Evaluarea în regăsirea informației	15
3.1	Evaluarea unui sistem RI	15
3.2	Colecții sandard de test	16
3.3	Evaluarea rezultatelor neordonate	16
3.4	Evaluarea rezultatelor ordonate	18
4	Metode de ierarhizare	23
4.1	Modelul spațiului vectorial (VSM)	23
4.1.1	Frecvența termenilor și ponderarea	23
4.1.2	Frecvența inversă a documentelor pentru un termen	23
4.1.3	Ponderarea <i>tf-idf</i>	24
4.1.4	Similaritatea cosinus	24
4.1.5	Variante de ponderare <i>tf-idf</i>	27
4.2	Euristici pentru eficientizare	28
4.2.1	Regăsirea inexactă a primelor K documente	28
4.2.2	Ignorarea termenilor	29
4.2.3	Listele de campioni	29
4.2.4	Scoruri statice	29
4.2.5	Impact ordering	30
4.2.6	Cluster pruning	31
4.3	Modelul probabilistic	31
4.3.1	Principiul probabilistic de ierarhizare	32
4.3.2	Modelul binar de independență	33
4.3.3	Okapi BM25	35
4.4	O abordare axiomatică	36
4.4.1	Spațiul funcțiilor	37
4.4.2	Constrângeri de regăsire	37
4.4.3	Modelarea funcțiilor	38

5	Metode de agregare	41
5.1	Borda	41
5.2	Agregarea Distanță-Rang	42
5.2.1	Distanța-rang	42
5.2.2	Problema rang-agregării	43
6	Studiu comparativ	45
6.1	Unelte	45
6.1.1	Apache Lucene	45
6.1.2	Proiectul Apache Open Relevance (ORP)	47
6.1.3	trec_eval	49
6.2	Framework-ul de test	50
6.3	Rezultate	52
6.3.1	VSM, BM25 și F2EXP	52
6.3.2	Preprocesarea textului	57
6.3.3	Cluster pruning	59
6.3.4	Îmbunătățirea performanțelor	61
6.3.5	Agregare	63

Capitolul 1

Introducere

Sintagma *regăsirea informației* (RI) are un spectru larg de înțelesuri. Scoaterea unui card de credit din portofel pentru a completa numărul cardului într-un formular este o formă de regăsire a informației. Dar, ca domeniu de studii academice, *regăsirea informației* poate fi definită astfel[1]:

Regăsirea informației se referă la găsirea de material (de obicei documente) de natură nestructurată (de obicei text) care satisface o nevoie de informație, din colecții mari de date (eventual stocate pe calculatoare).

Regăsirea informației obișnuia să fie o activitate practică de câțiva oameni cum ar fi bibliotecarii. Acum lumea s-a schimbat și sute de milioane de oameni folosesc sisteme de regăsire a informației precum motoare de căutare web în fiecare zi. Regăsirea informației devine rapid forma dominantă de accesare a informației întrecând modul tradițional de căutare de tip "baze de date" (căutarea după un număr de identificare).

RI este interdisciplinară, bazată pe informatică, matematică, știința informației, arhitectura informației, psihologie cognitivă, lingvistică, statistică, etc. Sisteme automate de RI au fost folosite pentru a reduce ceea ce se numește "supraîncărcarea de informație". Multe universități și librării publice folosesc sisteme RI pentru a facilita accesul la cărți, jurnale și alte tipuri de documente. Cele mai folosite aplicații RI sunt motoarele de căutare web.

1.1 Scurt istoric

Ideea de a folosi calculatoarele pentru a căuta "bucăți" de informație a fost popularizată în articolul *As We May Think* de către *Vannevar Bush* în anul 1945[4]. Primele sisteme automate de regăsire a informației au fost introduse în anii 1950 și 1960. Până în 1970 câteva tehnici au fost testate cu succes pe corpusuri mici de text cum ar fi colecția *Cranfield* (câteva mii de documente). Sisteme mari de RI cum ar fi *Lockheed Dialog system*, au fost date în folosință la începutul anilor 1970.

În 1992 Departamentul de Apărare al Statelor Unite împreună cu Institutul Național de Standarde și Tehnologie (NIST) au sponsorizat *TREC* (**T**ext **R**etrieval **C**onference) ca parte din programul TIP-STER. Scopul a fost să se asigure comunității RI infrastructura necesară pentru testarea și evaluarea metodelor de regăsire a textului pe colecții foarte mari. Acest program a acționat ca un catalizator în ceea ce privește cercetarea metodelor RI care se scalează la colecții foarte mari. Apariția motoarelor de căutare pe web a adus o nevoie și mai mare de sisteme RI care pot face față unor colecții imense de date.

1.2 Descrierea procesului

Un proces de regăsire a informației începe când un utilizator introduce o interogare (*query*) în sistem. Interogările sunt formulări formale ale *nevoilor de informație*, de exemplu, cuvinte scrise în caseta de căutare a unui motor de căutare web. În RI o interogare nu identifică în mod unic un obiect din colecția de obiecte. În schimb, mai multe obiecte pot fi considerate ca răspunsuri la interogare, eventual cu grade de relevanță diferite.

Un obiect este o entitate stocată în sistemul RI. În funcție de aplicație, obiectele pot fi texte, documente, imagini, videoclipuri, etc. În general, documentele propriu-zise nu sunt ținute în sistem ci sunt reprezentate de surrogate (rezultate în urma procesării documentelor) și metadate.

Majoritatea sistemelor RI calculează un scor numeric ce reprezintă cât de bine se potrivește un obiect cu interogarea utilizatorului și apoi ierarhizează obiectele în funcție de această valoare. Cele mai bune obiecte sunt apoi afișate utilizatorului.

1.3 Un exemplu

Să presupunem că avem la dispoziție o colecție de piese scrise de Shakespeare și vrem să determinăm care dintre aceste piese conține cuvintele *Brutus* și *Caesar* și *NU* conține cuvântul *Calpurnia*. O posibilitate este să se parcurgă fiecare piesă cuvânt cu cuvânt. Acest tip de scanare liniară este o formă foarte simplă de regăsire a documentelor și poartă numele de *grepping* (de la comanda Unix). Acest proces poate fi eficient, mai ales în contextul vitezei calculatoarelor moderne; pentru o colecție de dimensiuni reduse nu este nevoie de nimic mai mult.

Dar pentru o serie de alte scopuri, este nevoie de mai mult:

1. Procesarea rapidă a unei colecții mari de documente: cantitatea datelor on-line a crescut cel puțin la fel de repede ca viteza calculatoarelor, și se pune problema căutării în colecții care însumează trilioane de cuvinte.
2. Permitearea interogărilor flexibile.
3. Regăsirea ierarhizată: în multe cazuri, se dorește rezultatul cel mai bun pentru o nevoie de informație dintre mai multe documente care conțin anumite cuvinte.

Varianta alternativă la scanare liniară este indexarea anticipată a documentelor. Să presupunem că înregistrăm pentru fiecare document dacă conține sau nu fiecare cuvânt din setul total de cuvinte care apar în colecție. Rezultatul este o matrice binară de incidență numită *matrice termen-document* (figura 1.1). Termenii sunt unitățile indexate; sunt, de obicei, cuvinte, dar există cazuri când un termen este alcătuit din mai multe cuvinte (*Hong Kong*) sau este un cuvânt trunchiat. Este important să se facă diferențierea conceptuală dintre termen, ca unitate indexată, și cuvânt.

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Figura 1.1: Matrice termen-document pentru colecția pieselor lui Shakespeare. Elementul (t, d) este 1 dacă piesa din coloana d conține cuvântul din rândul t și 0, altfel.

Pentru a răspunde la interogarea *Brutus* și *Caesar* și *NU* *Calpurnia* se iau vectorii celor trei termeni, se completează ultimul, și se face operația binară *AND*:

$$110100 \text{ AND } 110111 \text{ AND } 101111 = 100100$$

Acest model se numește *modelul boolean de regăsire* și permite căutarea cu o interogare sub forma unei expresii boolene de termeni. Modelul boolean vede fiecare document ca un simplu set de cuvinte.

Scopul este proiectarea unui sistem adresat *regăsirii ad-hoc*. Acest tip de regăsire este cea mai întâlnită. În cadrul regăsirii ad-hoc, sistemul are drept scop să găsească documentele relevante la o

nevoie de informație aleatoare comunicată sistemului prin intermediul unei interogări. O *nevoie de informație* reprezintă subiectul despre care utilizatorul dorește să afle informații și diferă de *interogare* care este o încercare de a comunica nevoia de informație sistemului. Un document este *relevant* dacă utilizatorul îl percepe ca un set de informații "valoroase" cu privire la subiectul căutat. Pentru a evalua eficiența unui sistem RI (calitatea rezultatelor), un utilizator va dori să știe două valori statistice ale rezultatelor întoarse pentru o interogare:

- *Precizie*: Ce procent din rezultatele returnate sunt relevante la nevoia de informație?
- *Recall*: Ce procent din documentele relevante din colecție sunt returnate de sistem?

Mai multe detalii despre evaluarea unui sistem RI vor fi discutate în capitolul 3.

Pentru colecții foarte mari este inefficientă construirea unei matrice termen-document. Observația de bază este că matricea este foarte "risipită" (are foarte multe componente de zero). O reprezentare mai bună este înregistrarea ocurențelor (pozițiile de 1). Această idee e centrală pentru primul concept major din regăsirea informației: *indexul inversat*. Se păstrează un dicționar de termeni și, pentru fiecare termen, se stochează într-o listă referințele documentelor în care apare. Fiecare document din listă se numește *postare* și lista se numește *listă de postare*. În figura 1.2 se află un exemplu de index inversat.

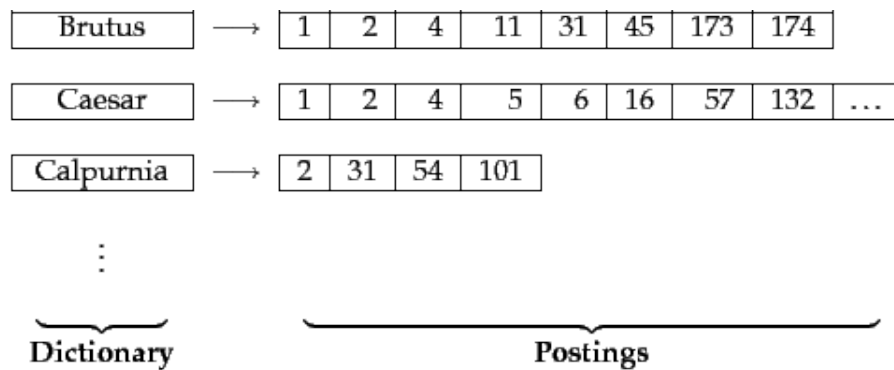


Figura 1.2: Două părți ale unui index inversat. Dicționarul este de obicei păstrat în memorie, în timp ce listele de postare sunt stocate pe disc.

Capitolul 2

Regăsirea informației - generalități

2.1 Procesul de indexare

Indexarea este procesul unui sistem de regăsire a informației prin care se colectează, transformă, și stochează informația pentru a facilita regăsirea rapidă și precisă. Proiectarea unui index presupune cunoștințe asupra unor concepte interdisciplinare din lingvistică, psihologie cognitivă, matematică, informatică, etc. Motoarele populare de căutare se concentrează pe indexarea de tip *full-text* ale documentelor online scrise în limbaj natural. Tipurile media(video, audio, imagini) pot fi și ele indexate. Motoarele de meta-căutare refolosesc indecșii altor servicii fără să stocheze un index local, în timp ce sistemele RI cu cache stochează permanent atât indexul cât și corpusul. Spre deosebire de indecșii full-text, un index *partial-text* restricționează "adâncimea" indexării pentru a reduce dimensiunea indexului. Unele servicii efectuează procesul de indexare la intervale predeterminate de timp din cauza costurilor de procesare, în timp ce alte motoarele de căutare indexează conținut în timp real.

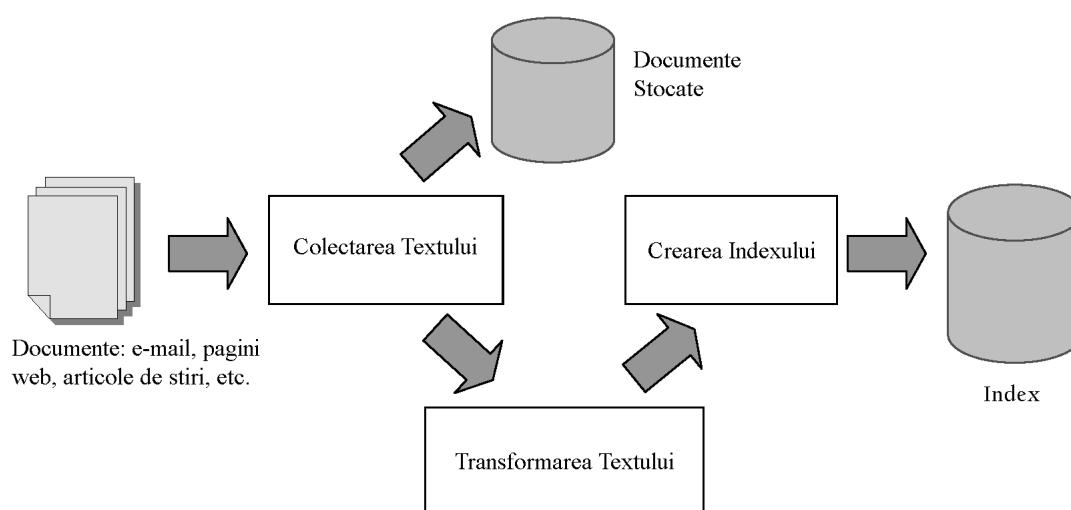


Figura 2.1: Indexarea

2.1.1 Colectarea textului

Una din principalele tehnici prin care un sistem RI colectează textul pentru indexare este procesul de *crawl-ing*. Un *crawler* indentifică și obține documente pe care motorul de căutare le va indexa. Există mai multe tipuri de *crawler-e*: *web*, *enterprise* și *desktop*. Crawler-ele web urmăresc hiperlegăturile dintr-un document pentru a găsi alte documente. Ele trebuie să furnizeze procesului de indexare un număr mare de pagini web, și, la eventuala modificare a acestora, trebuie să depisteze acest lucru și să

declanșeze reindexarea. Pentru asigurarea căutării pe un singur site, se folosesc crawler-ele de site care restricționează procesul de crawl-ing la un singur domeniu. Crawler-ele pentru căutarea de tip enterprise și desktop indexează conținut local din sisteme de management de conținut (CMS) respectiv harddisk-ul local. Un alt mod de colectare este reprezentat de *feed-uri*: stream-uri de documente în timp real (feed-uri web pentru știri, bloguri, video, radio, tv, etc.). Un standard pentru a transporta aceste feed-uri este RSS care poate furniza procesului de indexare documente XML.

După ce un document este colectat, înainte de indexare, acesta trebuie transformat într-un format consistent de text și metadata despre document. De exemplu, un motor de căutare care suportă mai multe formate (HTML, XML, Word, PDF), le poate transforma înainte de indexare într-un format XML. De asemenea, se adoptă și un standard al encodării (de exemplu, UTF-8) pentru noile formate.

Unele sisteme RI stochează documentele într-un *data store* înainte să le indexeze. Acest lucru facilitează accesul componentelor sistemului la setul de documente.

2.1.2 Transformarea textului

Înainte de a indexa un document acesta este împărțit în *tokeni* (parsat) pentru a identifica elementele structurale din text. Tokenii pot fi mai departe procesați prin diverse procedee cum ar fi *filtrarea*, *stemming-ul*, *lematizarea*. Tot în acest moment se pot efectua procese precum analiza legăturilor, extragerea informației și clasificare.

Parsarea

Dat fiind o secvență de caractere și o unitate de document, token-izarea este procesul de împărțire în bucăți numite "*token-i*" și, în același timp, eliminarea anumitor caractere precum semnele de punctuație. Acești token-i sunt numiți uneori termeni sau cuvinte, dar este important să se facă distincția între aceste concepte. Un token reprezintă instanța unei secvențe de caractere într-un anumit document, care sunt grupate într-o unitate semantică de care se va ține cont la procesare. Un *tip* este clasa tuturor token-ilor care conțin aceeași secvență de caractere. Un *termen* este un tip (de obicei normalizat) care este inclus în dicționarul sistemului RI. Termenii sunt strâns legați de token-ii din text, dar, de obicei, sunt obținuți prin aplicarea diferitelor procese de normalizare (de exemplu, îndepărtarea punctelor din prescurtări, eliminarea accentelor și diacriticelor sau transformarea tuturor literelor în litere mici), stemming și lematizare.

Întrebarea majoră care se pune în cadrul parsării este: *care sunt token-ii potriviți?* Raspunsul este că *acest lucru este specific limbii*. Așadar, limba documentului trebuie cunoscută. Identificarea limbajului bazată pe clasificatori care folosesc secvențe scurte de caractere drept caracteristici este foarte eficientă, majoritatea limbilor având modele distincte de astfel de semnături.

Filtrarea

Câteodată, unele cuvinte extrem de frecvente, care au o contribuție foarte mică în a ajuta la găsirea documentelor pentru o nevoie de informație, sunt excluse total din vocabularul de termeni. Aceste sunt numite *cuvinte de stop sau stop words*. Strategia generală pentru a alcătui o astfel de listă de filtrare este selectarea termenilor cu frecvențe foarte mari în colecția de documente. Folosirea unei liste de filtrare duce la reducerea semnificativă a numărului de postări din index și, de multe ori, nu are efecte negative (interogările de tip frază exactă constituie un exemplu în care lista de stop are efecte negative).

Iată o listă de cuvinte de stop pentru limba engleză:

a, an, and, are, as, at, be, by, from, for, has, he, in, is, it, its, on, of, that, the, to, was, where, will, with

Tendința folosirii listelor de stop de-a lungul timpului a scăzut de la liste foarte mari (200-300 termeni), la liste foarte mici (7-12 termeni), ajungându-se la a nu se folosi deloc. În general, motoarele de căutare web nu folosesc cuvinte de stop ci întrebuințează statistici ale limbii pentru tratarea mai bună a cuvintelor comune.

Stemming și lematizare

Documentele folosesc diferite forme ale aceluiași cuvânt, de exemplu: *organizez*, *organizează*, *organizând*. În multe situații pare util ca o căutare a unuiu dintre aceste cuvinte să întoarcă documente care folosesc oricare din formele lui.

Atât scopul *stemming*-ului cât și al *lematizării* este *reducerea formelor flexionare și a formelor derivate cu semantică similară a unui cuvânt la o formă de bază*. De exemplu:

am, are, is \Rightarrow be

car, cars, car's, cars' \Rightarrow car

Rezultatul aplicării acestui tip de reguli este următorul:

the boy's cars are different colors \Rightarrow the boy car be differ color

Deși au un scop comun, cele două concepte diferă prin mijloacele folosite. Stemming-ul se referă de obicei la procese euristice folosite pentru a îndepărta terminațiile cuvintelor în speranța atingerii scopului de cele mai multe ori. Lematizarea încearcă să atingă scopul prin mijloace mai "formale", folosind un vocabular și analizând morfologic cuvintele (forma din dicționar a unui cuvânt poartă numele de *lemă*). Confruntate cu secvența de caractere *saw*, un proces de stemming ar putea întoarce *s*, în timp ce lematizarea ar încerca să întoarcă *see* sau *saw* în funcție de rolul cuvântului (verb, respectiv substantiv).

Cel mai comun algoritm de stemming în engleză este *algoritmul Porter*. S-a constatat în mod empiric faptul că este foarte eficient.

2.1.3 Crearea indexului

Scopul indexării este adunarea de "statistici" despre documente și stocarea lor într-o structură ușor accesibilă. Aceste statistici pot fi, de exemplu, frecvență și pozițiile cuvintelor. De obicei, aceste statistici sunt combinate în ponderi precum *tf-idf* (secțiunea 4.1.3) folosite în de algoritmul de ierarhizare. Tehnica de bază a indexării este *inversiunea*. Aceasta are rolul de a converti informație de tip document-termen în termen-document pentru accesarea rapidă la interogare. Structura de date folosită trebuie să fie capabilă să suporte actualizări frecvente. Pentru a minimiza costurile de stocare se utilizează *compresia* care are ca beneficiu și facilitarea folosirii sistemelor de caching (mai multă informație poate încăpea în memorie). Structura folosită de un sistem RI este *indexul inversat*. Acesta este o structură de date care stochează o mapare de la conținut (în acest caz, cuvinte) la locația acestuia (documente). Date fiind textele $T_0 = \text{"it is what it is"}$, $T_1 = \text{"what it is"}$ și $T_2 = \text{"it is a banana"}$, un index inversat are următoarea formă:

"a"	[2]
"banana"	[2]
"is"	[0, 1, 2]
"it"	[0, 1, 2]
"what"	[0, 1]

Tabelul 2.1: Index inversat

O căutare după termenii *"what"*, *"is"* și *"it"* ar avea următoarele rezultate:

$$\{0, 1\} \cup \{0, 1, 2\} \cup \{0, 1, 2\} = \{0, 1\}.$$

Pentru aceleași texte se pot stoca și pozițiile pe care se găsesc termenii (tabelul 2.2).

2.2 Procesul de căutare

2.2.1 Interacțiunea cu utilizatorul

Un sistem RI pune la dispoziția utilizatorului o interfață prin intermediul căruia se poate interoga indexul de documente. În general, această interfață este destul de simplă, constând într-o casetă în care

"a"	$[(2, 2)]$
"banana"	$[(2, 3)]$
"is"	$[(0, 1), (0, 4), (1, 1), (2, 1)]$
"it"	$[(0, 0), (0, 3), (1, 2), (2, 0)]$
"what"	$[(0, 2), (1, 0)]$

Tabelul 2.2: Index pozițional

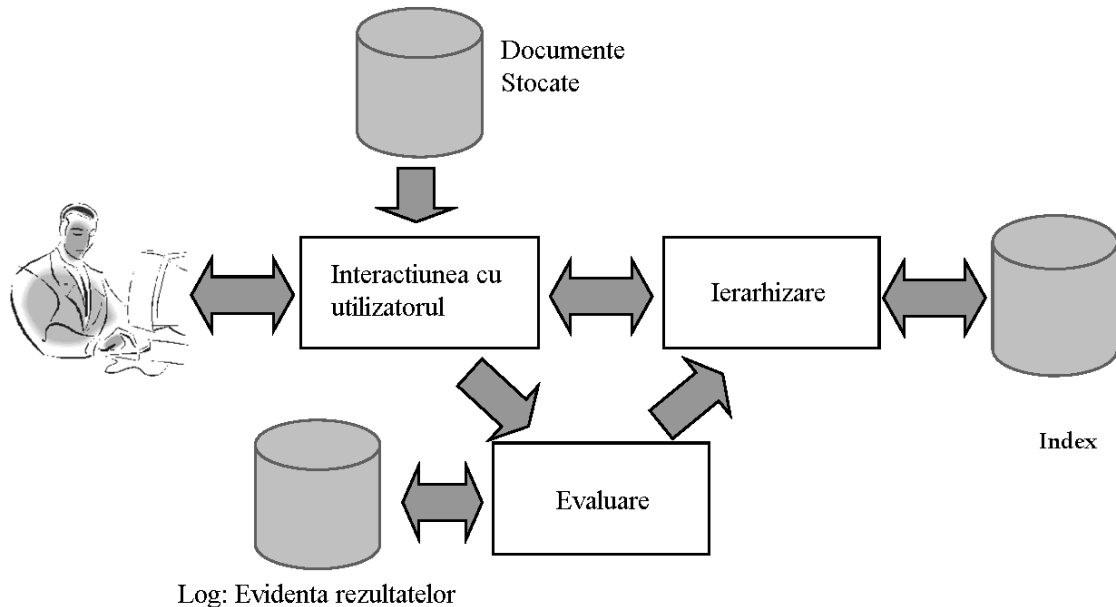


Figura 2.2: Interogarea indexului

utilizatorul scrie interogarea și un buton pentru lansarea căutării. Interogarea este apoi procesată de un parser care interpretează limbajul folosit. Sintaxa interogărilor diferă în funcție de sistem, dar de obicei (mai ales pe web) se folosesc interogări de tip free-text și eventual câțiva operatori logici. După ce interogarea este trasformată într-o reprezentare internă, se aplică aceleași transformări ale textului aplicate documentului (filtrare, normalizare, stemming). Unele sisteme verifică scrierea corectă și pot indica sugestii în cazul detectării erorilor. Pentru a rafina scopul căutării, un motor de căutare poate extinde interogarea prin adăugarea de termeni (sinonime, concepte înrudite).

După ce căutarea și ierarhizarea au avut loc, interfața afișează lista de documente. Pentru fiecare document din listă se afișează și fragmente în care termenii au fost regăsiți (eventual termenii sunt scoși în evidență). Anumite sisteme afișează și reclame relevante la interogarea respectivă și anumite unelte de vizualizare pentru rafinarea interogării sau efectuarea de interogări înrudite.

2.2.2 Procesul de regăsire și ierarhizare

Algoritmul de ierarhizare poate fi considerat componenta de bază a unui sistem de regăsire a informației. Strategiile de regăsire au scopul de a asigura un scor de relevanță între o interogare și un document. Aceste strategii sunt bazate pe noțiunea conform căreia cu cât un termen din interogare apare mai des în document, cu atât este mai "relevant" documentul la interogare. Unele dintre aceste tehnici aplică anumite măsuri care să atenueze problemele cauzate de ambiguitățile limbii (același concept poate fi exprimat folosind termeni diferiți sau același termen poate avea sensuri diferite în contexte diferite). Un algoritm de ierarhizare are la intrare o interogare Q și un set de documente D_1, D_2, \dots, D_n iar scopul lui este să identifice un coeficient de similaritate $SC(Q, D_i)$ pentru $1 \leq i \leq n$ (SC - *similarity coefficient* - este uneori numit RSV - *retrieval status value*).

Există mai multe modele de regăsire și ierarhizare. O parte din ele sunt ilustrate în figura 2.3 și câteva dintre acestea sunt descrise pe scurt în paragrafele următoare[2].

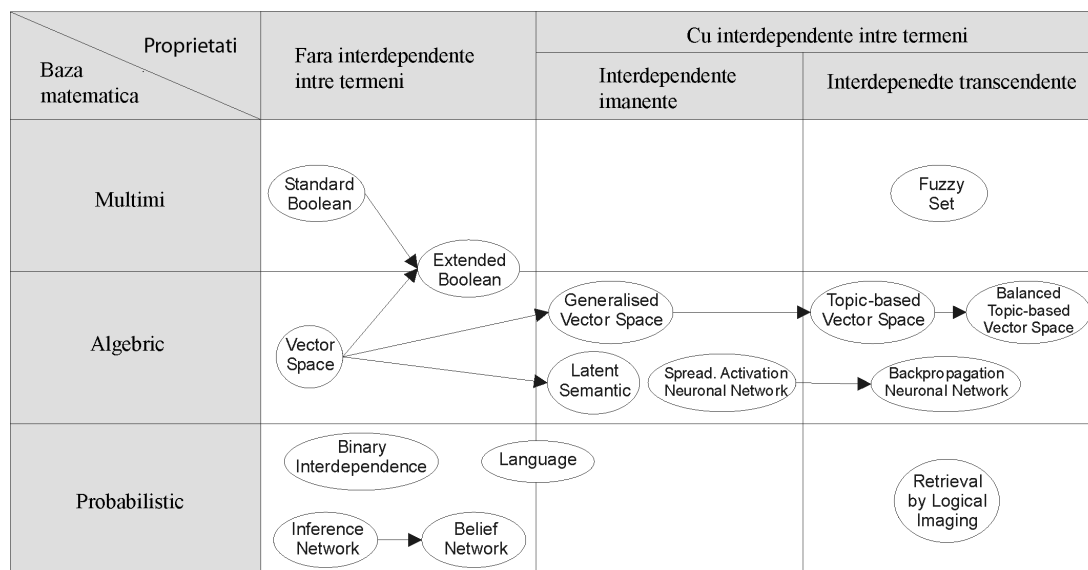


Figura 2.3: Diferite modele de ierarhizare clasate după metoda folosită și modul de tratare a termenilor[4]

Modelul boolean La căutare sunt întoarse documentele care satisfac condiția logică impusă de interogare și, de obicei, se ține cont doar de apariția respectiv lipsa unui termen din document, nu și de frecvență.

Modelul spațiului vectorial În cadrul acestui model, atât interogarea cât și documentele sunt reprezentate ca vectori într-un spațiu vectorial cu câte o dimensiune pentru fiecare termen din colecție. O măsură de similaritate este calculată între interogare și un document ca un produs scalar între cei doi vectori normalizați, sau o variațiune a acestuia.

Regăsirea probabilistică Pentru fiecare termen din colecție este calculată probabilitatea ca acesta să apară într-un document relevant. Similaritatea dintre un document și o interogare este calculată ca o combinație a probabilităților termenilor comuni.

Modele lingvistice Un model lingvistic este calculat pentru fiecare document și, la căutare, se calculează probabilitatea ca un document să genereze interogarea.

Rețele de inferență O rețea bayesiană este folosită pentru a infera relevanța unui document la o interogare. Această concluzie se trage pe baza "evidențelor" din document.

Indexarea semantică latentă Ocurențele termenilor în documente sunt reprezentate folosind o matrice *termen-document*. Matricea este redusă prin SVD (*Singular Value Decomposition*) pentru a filtra "gălăgia" (*noise*) găsită în documente astfel încât două documente care au semantici similare să fie situate cât mai aproape într-un spațiu multi-dimensional.

Rețele neurale Anumiți "neuroni" dintr-o secvență (noduri într-o rețea) sunt activați de o interogare și produc legături către documente. "Puterea" fiecărei legături din rețea este transmisă documentului și colectată pentru a forma un coeficient de similaritate între interogare și document. Rețelele sunt antrenate prin ajustarea ponderilor legăturilor pe colecții cu interogări și răspunsuri predeterminate.

Algoritmi genetici O interogare optimă pentru a găsi documente relevante la o nevoie de informație poate fi generată prin evoluție. O interogare inițială este folosită cu ponderi aleatoare sau estimate ale termenilor. Noi interogări sunt generate prin modificarea acestor ponderi. O nouă interogare "supraviețuiește" dacă este similară cu documentele care se știu a fi relevante la nevoia de informație (predeterminate).

Regăsirea pe mulțimi fuzzy Un document este mapat la o mulțime fuzzy (o mulțime în care fiecare element are asociat un număr care indică "puterea de apartenență"). Interogările boolene sunt mapate la intersecții, reuniuni și complemente de mulțimi fuzzy care au ca rezultat o pondere de apartenență asociată cu fiecare document care e relevant la interogare. Această pondere este folosită ca măsură de similaritate.

Pentru fiecare strategie se pot folosi diferite utilități pentru a-i îmbunătăți rezultatele. Acestea pot fi: *expandarea interogărilor*, *feedback-ul de relevanță*, *clustering*, *n-grame*, *rețele semantice*, etc. Încercând să rafineze interogarea, majoritatea acestor utilități adaugă sau îndepărtează termeni de la interogarea inițială. Aceste utilități sunt componente "*plug-and-play*" în sensul că ar trebui să poată fi folosite cu orice model de ierarhizare.

O tratare mai detaliată a câtorva dintre aceste metode se va face în capitolul 4.

2.2.3 Evaluarea

Pentru îmbunătățirea eficienței sistemului este crucială stocarea interogărilor și interacțiunilor într-un log. Analiza eficienței ierarhizării permite ajustarea sistemului pentru a da rezultate cât mai relevante la nevoile de informație și constituie tema de bază a acestei lucrări. Tehnicile de evaluare sunt descrise în capitolul 3.

Capitolul 3

Evaluarea în regăsirea informației

Există multe alternative pentru a proiecta un sistem RI. Cum hotărâm care dintre aceste tehnici sunt eficiente în anumite aplicații? Este indicată folosirea unui *stop list* sau a unui *stemmer*? Regăsirea informației a evoluat ca o disciplină empirică, necesitând evaluări atente și complete pentru a demonstra superioritatea performanțelor noilor tehnici apărute pe colecții reprezentative de documente.

3.1 Evaluarea unui sistem RI

Pentru a măsura eficiența unui sistem ad-hoc RI este necesară o colecție de test alcătuită din trei componente:

1. O colecție de documente
2. O serie de *nevoi de informație* exprimate prin interogări
3. Un set de judecăți de relevanță, de obicei o valoare binară (*relevant/nerelevant*) pentru fiecare pereche interogare-document.

Abordarea standard în evaluarea unui sistem RI se învârtă în jurul noțiunii de documente *relevante* și *nerelevante*. Cu privire la o nevoie de informație, unui document dintr-o colecție de test i se dă o clasificare binară: relevant sau nerelevant. Colecția de documente și setul de interogări trebuie să fie destul de mari: este nevoie de efectuarea unei medii a rezultatelor de performanță peste seturi mari de test întrucât acestea variază destul de mult de la interogare la interogare. Ca o regulă derivată din experimente, 50 de nevoi de informație reprezintă un minim suficient.

Relevanța este evaluată relativ la nevoia de informație, nu la interogare. De exemplu, o nevoie de informație ar putea fi:

Informație cu privire la faptul că vinul roșu este mai eficient în prevenirea atacurilor de cord decât vinul alb.

Interogarea asociată acestei nevoi de informație ar putea fi:

vin ȘI roșu ȘI alb ȘI atac ȘI cord ȘI eficient

Un document este relevant dacă se adresează nevoii de informație nu doar pentru că se întâmplă să conțină cuvintele din interogare. Acest lucru este des neînțeles în practică, deoarece nevoia de informație este "ascunsă". În orice caz, ea este prezentă. Pentru o interogare care constă într-un cuvânt, este foarte greu pentru un sistem RI să își dea seama care este nevoia de informație. Dar utilizatorul are o astfel de nevoie și va judeca rezultatele pe baza acesteia. Pentru a evalua un sistem este nevoie de o exprimare "deschisă" a nevoii de informație, care poate fi folosită la judecarea documentelor întoarse de sistem ca relevante sau nerelevante. Relevanța poate fi gândită ca o scală, cu unele documente *mai* relevante decât altele. Dar, pentru început, se va folosi o valoare binară a relevanței.

Multe sisteme conțin diferiți parametri de calibrare ce pot fi ajustați pentru a mări performanța. Este greșit să se raporteze rezultate pe o colecție de test obținute în urma modificării acestor parametri cu scopul de a maximiza performanța pe colecția respectivă deoarece parametrii vor fi setați să maximizeze performanța pe un anumit set de interogări în loc să facă acest lucru pentru un eșantion aleator de interogări.

3.2 Colecții standard de test

Aceasta este o listă de colecții standard de test și evaluare. Majoritatea sunt menite pentru a testa sisteme ad-hoc de RI dar sunt și câteva folosite pentru clasificare de text.

Cranfield Această colecție a fost prima care a permis măsuri precise de eficiență ale sistemelor de regăsire a informației, dar acum este prea mică pentru un experiment "serios". A fost alcătuită în anii 1950 în Regatul Unit și conține 1398 de sumare de articole despre aerodinamică, un set de 225 de interogări și judecăți de relevanță complete pentru toate perechile (interogare, document).

TREC NIST a construit un framework de evaluare pentru RI începând cu 1992. Din acest framework cele mai cunoscute colecții de test sunt cele folosite la testele *TREC Ad Hoc track* în timpul primelor opt conferințe TREC (**T**ext **R**etrieval **C**onference) între 1992 și 1999. În total, aceste colecții de test însumează 6 CD-uri și conțin 1,89 milioane de documente (compuse cu preponderență din articole de știri) și judecăți de relevanță pentru 450 de nevoi de informație numite "subiecte". Colecțiile TREC 6-8 sunt formate din 150 de nevoi de informație și 528.000 de articole de știri. Acesta este probabil cea mai bună subcolecție datorită faptului că este cea mai mare și subiectele sunt mai consistente. Din cauza faptului că aceste colecții sunt foarte mari, nu există judecăți exhaustive ci sunt specificate doar pentru un subset de documente (primele k documente întoarse de sistemul pentru care nevoia de informație a fost dezvoltată).

GOV2 În ultimii ani, NIST a făcut evaluări pe colecții mult mai mari, incluzând colecția de 25 de milioane de pagini web, *GOV2*. GOV2 este acum cea mai mare colecție de pagini web disponibilă pentru cercetare. Cu toate acestea, GOV2 încă este de cel puțin 2 ori mai mică decât colecția de documente indexate de companiile mari de căutare web.

NTCIR Proiectul *NTCIR - NII Test Collections for IR Systems* a construit diferite colecții de test de mărimi similare cu colecțiile TREC. Aceste colecții sunt concentrate pe limba est-asiatică și pe *regăsirea informației cross-language* (interogările sunt făcute într-o limbă peste un set de documente scrise în diferite limbi).

CLEF Seria de evaluări CLEF (Cross Language Evaluation Forum) s-a concentrat pe limbile europene și regăsirea informației de tip cross-language.

Reuters Pentru clasificarea de text, cea mai folosită colecție de test a fost *Reuters-21578* compusă din 21578 articole de știri. Mai recent Reuters a publicat *Reuters Corpus Volume 1 (RCV1)*, o colecție mult mai mare constând 806.791 documente.

3.3 Evaluarea rezultatelor neordonate

Având aceste ingrediente, cum este măsurată eficiența unui sistem? Cele mai frecvente și fundamentale două măsuri în RI sunt *precizia* (*precision* - P) și *recall-ul* (R). Acestea sunt mai întâi definite pentru cazul în care sistemul RI întoarce un set neordonat de documente pentru o interogare și apoi extinse la liste ierarhizate de documente.

Precizia este dată de raportul dintre documentele regăsite care sunt relevante și documentele regăsite:

$$P = \frac{\#(\text{documente regăsite și relevante})}{\#(\text{documente regăsite})} = P(\text{relevante}|\text{regăsite}) \quad (3.1)$$

Recall-ul este raportul dintre documentele regăsite care sunt relevante și documentele relevante:

$$R = \frac{\#(\text{documente regăsite și relevante})}{\#(\text{documente relevante})} = P(\text{regăsite}|\text{relevante}) \quad (3.2)$$

Aceste noțiuni pot fi explicate mai clar folosind tabelul de contingență de mai jos.

	Relevante	Nerelevante
Regăsite	pozitive adevărate(tp)	pozitive false(fp)
Neregăsite	negative false(fn)	negative adevărate(tn)

Atunci:

$$P = \frac{tp}{tp + fp} \quad R = \frac{tp}{tp + fn} \quad (3.3)$$

O alternativă evidentă este judecarea sistemului după acuratețe, adică, procentul clasificărilor corecte. Folosind tabelul de mai sus, $ac = (tp + tn)/(tp + fp + fn + tn)$. Acest lucru pare plauzibil din moment ce există două clase, relevant și nerelevant, iar un sistem RI poate fi privit ca un clasificator pe două clase (întoarce documentele etichetate cu *relevant*). Aceasta este, de fapt, măsura de eficiență folosită de obicei în evaluarea problemelor de clasificare.

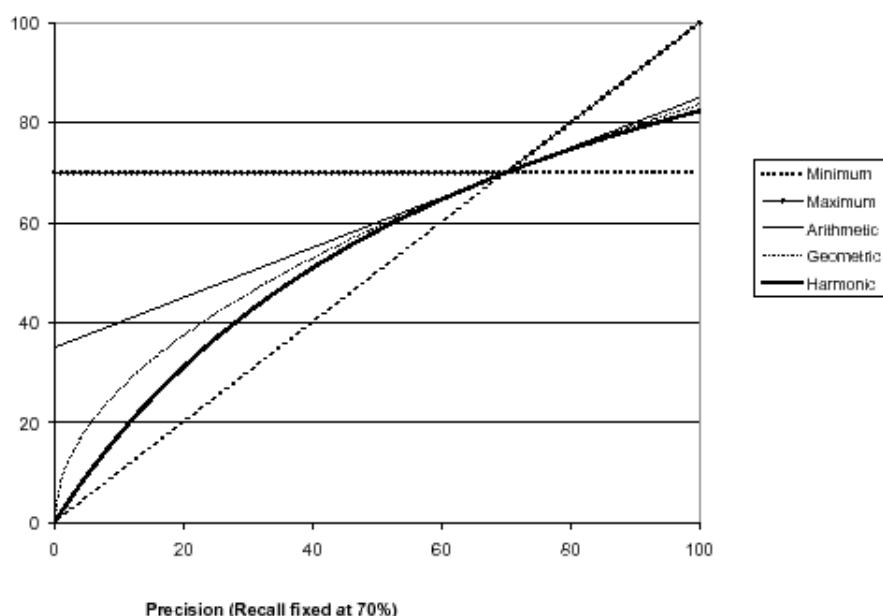


Figura 3.1: Grafic care compară media armonică cu celelalte medii

Există un motiv din cauza căruia acuratețea nu este o măsură corespunzătoare pentru regăsirea informației. În aproape toate circumstanțele, datele sunt extrem de "denaturate": de obicei, peste 99,9% din documente sunt în categoria nerelevant. Un sistem configurat să maximizeze acuratețea poate părea că funcționează foarte bine prin considerarea tuturor documentelor ca fiind nerelevante la orice întrebare. Acest lucru este inacceptabil pentru un sistem RI. Un utilizator va dori întotdeauna să vadă niste documente și poate avea o mică toleranță pentru pozitive false dacă majoritatea documentelor îi satisfac nevoia de informație. Măsurile P și R concentrează evaluarea pe întoarcerea de pozitive adevărate, întrebând ce procent de documente relevante a fost găsit și cate pozitive false au fost întoarse.

Avantajul faptului că există două măsuri diferite este acela că în cele mai multe circumstanțe una este mai importantă decât cealaltă. Un utilizator web obișnuit dorește ca fiecare rezultat de pe prima pagină să fie relevant (precizie mare) dar nu are nici cel mai mic interes să știe și mai ales să privească fiecare document relevant. Pe de altă parte, un profesionist care caută documente într-un sistem RI intern va încerca să găsească toate documentele relevante la nevoia lui de informație (recall mare) și va tolera valori mici ale preciziei pentru a își atinge scopul. În orice caz, există un compromis între cele două valori: se poate întotdeauna obține un recall maxim dar precizie mică dacă se întorce o listă cu toate documentele. Recall-ul este o funcție non-descrescătoare de numărul de documente regăsite. Pe de altă parte, într-un sistem bun, precizia de obicei scade odată cu creșterea numărului de documente regăsite. În general se dorește o anumită valoare a recall-ului, tolerându-se un anumit procent de pozitive false.

O măsură care înglobează atât precizia cât și recall-ul este *măsura F*, care se calculează ca media

armonică ponderată a celor două valori:

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \text{ unde } \beta^2 = \frac{1 - \alpha}{\alpha} \quad (3.4)$$

unde $\alpha \in [0, 1]$ și, ca urmare, $\beta^2 \in [0, \infty)$. Măsura F balansată ține cont în mod egal de precizie și recall ($\alpha = 1/2, \beta = 1$). Este notată de obicei cu F_1 (de la $F_{\beta=1}$):

$$F_{\beta=1} = \frac{2PR}{P + R} \quad (3.5)$$

Valori ale lui β mai mici decât 1 scot în evidență precizia, în timp ce valori mai mari ca 1 pun accentul pe recall. Valorile discutate până acum sunt măsuri între 0 și 1, dar se mai scriu și ca procente pe o scală de la 1 la 100. Media armonică este folosită în locul mediei aritmetice deoarece în cazul în care un sistem ar întoarce toate documentele (100% recall), ar avea tot timpul cel puțin $F = 50\%$. Media armonică a două numere ($a \ll b$) este mai apropiată de minim decât de media lor aritmetică (vezi figura 3.1).

3.4 Evaluarea rezultatelor ordonate

Precizia, recall-ul și F sunt bazate pe seturi; sunt calculate folosind seturi neordonate de documente. Este nevoie de o extindere a acestor măsuri (sau de o definiție de noi măsuri) dacă se dorește evaluarea unor rezultate ierarhizate care reprezintă standardul în zilele noastre. Într-un context de regăsire ierarhizată setul care interesează este dat de *primele k* documente. Pentru fiecare astfel de set, valorile preciziei și recall-ului pot fi reprezentate grafic printr-o *curbă precizie-recall*, ca cea din figura 3.2.

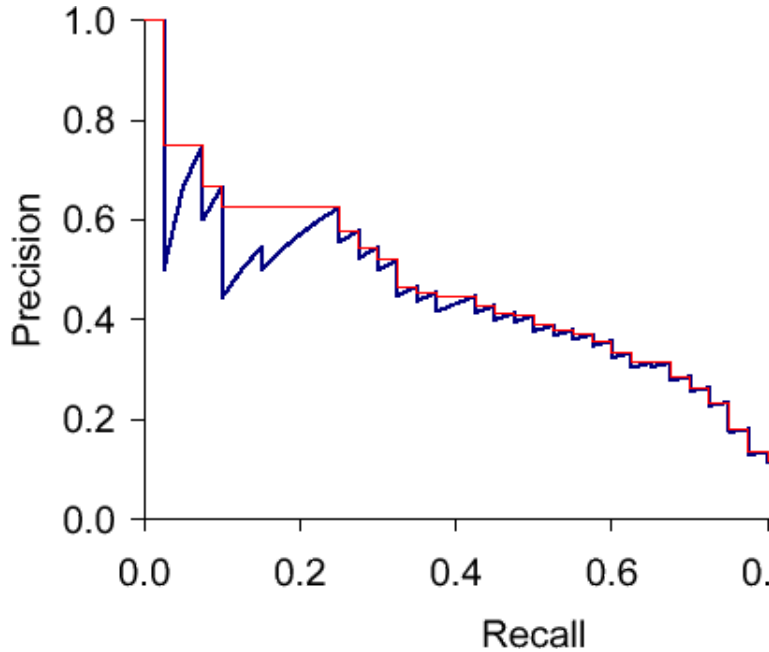


Figura 3.2: Grafic *precizie-recall*.

Curbele precizie-recall au o formă de "fierăstrău": dacă al $(k+1)$ -lea document regăsit este nerelevant, atunci recall-ul este același ca pentru primele k documente regăsite, dar precizia scade. Dacă documentul este relevant atunci ambele valori cresc. Este de obicei util să se elimine această formă și modul standard de a face acest lucru este folosind *precizia interpolată* p_{interp} . La un anumit nivel de recall r , p_{interp} este dat de cea mai mare valoare a preciziei găsite pentru orice nivel de recall $r' \geq r$:

$$p_{interp} = \max_{r' \geq r} p(r') \quad (3.6)$$

Precizia interpolată este ilustrată în figura 3.2.

Examinarea întregii curbe precizie-recall este foarte informativă, dar, câteodată este necesar ca informația să fie redusă la câteva numere. Modul tradițional de a realiza acest lucru (folosit, de exemplu, la primele 8 evaluări TREC Ad Hoc) este *precizia medie interpolată în 11 puncte*. Pentru fiecare nevoie de informație, precizia interpolată este măsurată în cele 11 puncte de recall: 0.1, 0.2, ..., 1.0. Exemplul din figură 3.2 este ilustrat în tabelul 3.1.

Recall	Precizie interpolată
0.0	1.00
0.1	0.67
0.2	0.63
0.3	0.55
0.4	0.45
0.5	0.41
0.6	0.36
0.7	0.29
0.8	0.13
0.9	0.10
1.0	0.08

Tabelul 3.1: Calcularea preciziei medii interpolate în 11 puncte.

Pentru fiecare nivel de recall se calculează media aritmetică a preciziei interpolate la acel nivel de recall pentru fiecare nevoie de informație din colecție. O curbă precizie-recall cu 11 puncte poate fi vizualizată apoi grafic. Un exemplu este prezent în figura 3.3.

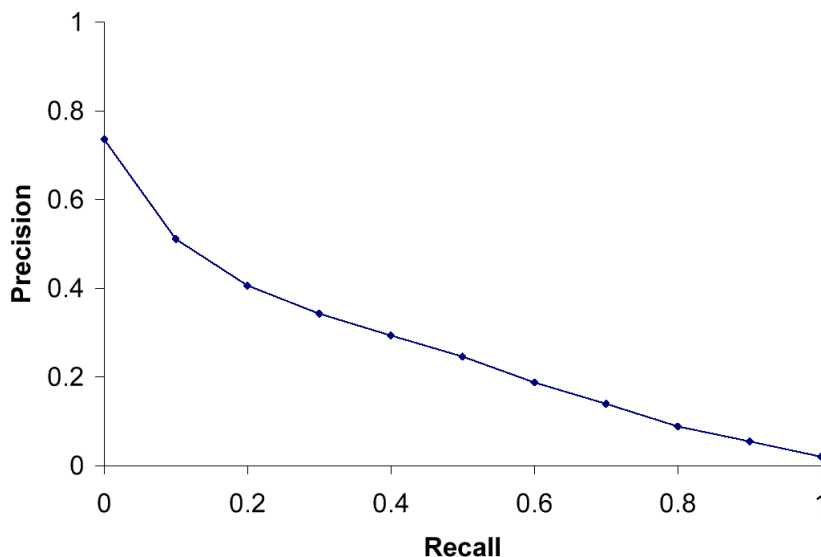


Figura 3.3: Grafic al preciziei medii interpolate în 11 puncte.

În ultimii ani, alte măsuri au devenit mai comune. Cea mai folosită în comunitatea TREC este *Mean Average Precision (MAP)*, care pune la dispoziție o singură măsură de calitate de-a lungul nivelelor de recall. Printre tipurile de măsuri de evaluare s-a arătat că MAP funcționează foarte bine la capitolele discriminare și stabilitate. Pentru o singură nevoie de informație, precizia medie este media valorilor de precizie obținute pentru primele k documente după regăsirea fiecărui document relevant. Media acestei valori peste toate nevoile de informație este MAP. Dacă setul de documente relevante pentru o nevoie de informație $q_j \in Q$ este d_1, \dots, d_{m_j} și R_{jk} este setul de rezultate de la primul până la documentul d_k , atunci:

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Prec(R_{jk}) \quad (3.7)$$

Când un document relevant nu este regăsit, precizia este considerată 0. Pentru o singură nevoie de

informație, precizia medie aproximează aria de sub curba neinterpolată precizie-recall, așadar, MAP aproximează aria medie de sub curba precizie-recall pentru un set de interogări.

Folosind MAP, nu sunt alese nivele fixe de recall și nu este nevoie de interpolare. Un set de nevoi de informație trebuie să fie cuprinzător și diversificat pentru a putea măsura cât mai exact eficiența unui sistem.

Măsurile descrise până acum au în componență precizia pentru fiecare nivel de recall. În cazul multor aplicații, cum ar fi un motor de căutare web, acest lucru nu reflectă neapărat nevoile unui utilizator. În acest caz contează mai mult câte rezultate bune se găsesc în primele pagini. Acest lucru conduce la nevoia de a măsura precizia pentru primele documente regăsite (10 sau 30). Această tehnică poartă numele de "*Precizia la k*". Are avantajul că nu necesită o estimare a dimensiunii setului de documente relevante (m_j în cazul MAP) și dezavantajele că este cea mai puțin stabilă dintre metodele de evaluare și că nu se poate calcula media prea bine pentru ea.

O alternativă care atenuează această problemă este *R-precizia*. Implică existența unui set *Rel* de documente despre care se știe că sunt relevante, din care se calculează apoi precizia pentru primele $|Rel|$ documente regăsite. R-precizia se ajustează la mărimea setului de documente relevante: un sistem perfect ar putea puncta 1 la această măsură pentru fiecare interogare, în timp ce până și un sistem perfect, ar putea să atingă o precizie la 20 de 0.4 dacă ar exista doar 8 documente în colecția de documente relevante. Calcularea mediei peste setul de interogări are mai mult sens în acest caz. Dacă există $|Rel|$ documente relevante pentru o interogare, se examinează primele $|Rel|$ rezultate date de un sistem și se găsesc r documente relevante, atunci precizia (și deci R-precizia) este $r/|Rel|$, dar și recall-ul este tot $r/|Rel|$. Așadar, R-precizia este identică cu o altă măsură folosită câteodată: *break-even point*, măsură definită ca valoarea la care precizia și recall-ul sunt egale. Ca și *Precizia la k*, *R-precizia* descrie un singur punct de pe curbă, și e uneori neclar de ce interesează mai mult nivelul în care cele două valori sunt egale decât nivelul cel mai bun de pe curbă (în care F este maxim) sau un nivel de interes pentru o anumită aplicație (*Precizia la k*). Cu toate acestea, R-precizia pare să fie corelată cu MAP, lucru constatat în urma experimentelor.

Un alt concept folosit uneori în evaluarea unui sistem este *curba ROC* ("*ROC*" vine de la *Receiver Operating Characteristics*). O curbă ROC reprezintă grafic rata pozitivelor adevărate (*sensitivitate*) ca funcție de rata pozitivelor false ($1 - \text{specificitate}$). Aici sensibilitate este un alt termen pentru recall. Rata de pozitive false este dată de $fp/(fp + tn)$. Figura 3.4 ilustrează curba ROC corespunzătoare curbei precizie-recall din figura 3.2.

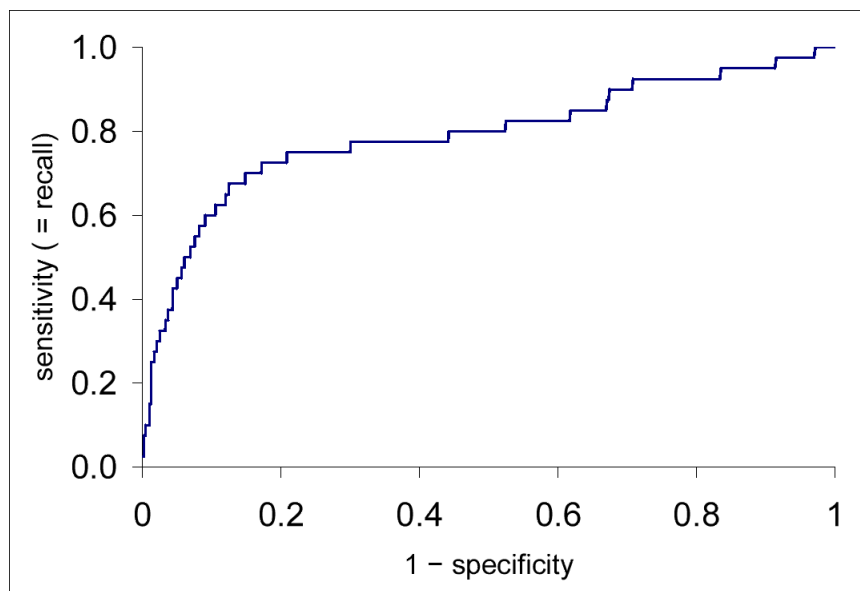


Figura 3.4: Curba ROC

O curbă ROC urcă întotdeauna din stânga jos spre dreapta sus. Pentru un sistem bun, graficul urcă abrupt în partea stângă. Are mai mult sens când se ia în considerare întreg spectrul de regăsire și pune la dispoziție o altă perspectivă asupra datelor. În unele cazuri se folosește estimarea ariei de sub curba

ROC, care este o valoare analoagă măsurii MAP.

O altă abordare care a fost adoptată din ce în ce mai des, în special în cazul sistemelor care folosesc *machine learning*, este măsura *câștigului cumulativ* (*cumulative gain*) și, în particular, *normalized discounted cumulative gain* (*NDCG*). NDCG este proiectat pentru situații în care relevanța este nebinară. Ca și *Precizia la k*, este evaluată peste primele k rezultate. Pentru un set de interogări Q, fie $R(j, d)$ scorul de relevanță pe care evaluatorii l-au dat documentului d pentru interogarea j. Atunci,

$$NDCG(Q, k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^k \frac{2^{R(j,m)} - 1}{\log_2(1 + m)}, \quad (3.8)$$

unde Z_{kj} este un factor de normalizare calculat pentru a face ca NDCG-ul unei ierarhizări perfecte la k pentru interogarea j să fie 1. Pentru interogări pentru care $k' < k$ documente sunt returnate, ultima sumă se face până la k'

Capitolul 4

Metode de ierarhizare

4.1 Modelul spațiului vectorial (VSM)

4.1.1 Frecvența termenilor și ponderarea

Spre deosebire de o interogare booleană, o interogare *liberă* ("free text") e definită de un set de cuvinte neconectate de operatori. Acest tip de interogare este foarte populară pe web. Un mecanism de scor pentru această situație poate fi calcularea sumei dintre scorurile pe care le obține fiecare termen din interogare cu un document.

În acest scop, se asignează fiecărui termen din document o pondere care depinde de numărul de ocurențe ale termenului în document. Se vrea calcularea unui scor între termenul unei interogări, t și un document d , pornind de la ponderea lui t în d . Cea mai simplă abordare este ca ponderea să fie chiar frecvența termenului în document. Această schemă de ponderarea este numită *term frequency* și notată $tf_{t,d}$.

Pentru un document d , setul de ponderi tf (sau orice alte ponderi care mapează frecvența termenilor la numere reale pozitive) poate fi văzut ca o reprezentare a documentului. În această perspectivă asupra unui document, cunoscută în literatură sub numele de *modelul bag of words*, ordinea cuvintelor în document este ignorată, dar, spre deosebire de modelul boolean, se ține cont de frecvența lor. În orice caz, pare intuitiv faptul că două documente care au reprezentări similare, sunt similare și în conținut.

O altă dilemă care apare este următoarea: toate cuvintele dintr-un document sunt la fel de importante? În mod evident, nu. Pe lângă conceptul de *cuvinte de stop* - cuvinte care nu se indexează, și ca urmare nu contribuie la calcularea scorului - există alt mecanism pentru a cântări importanța cuvintelor: *frecvența inversă a documentelor pentru un termen* (*idf* - *inverse document frequency*).

4.1.2 Frecvența inversă a documentelor pentru un termen

Problema cu ponderarea tf este că unii termeni ar trebui să aibă mai puțină putere de discriminare decât alții în determinarea relevanței. Este nevoie de un mecanism care să atenueze efectul termenilor care au o frecvență mare la nivel de colecție. O primă idee este reducerea ponderii tf a unui termen cu un factor proporțional cu frecvența la nivel de colecție (numărul de ocurențe în toate documentele).

Cuvânt	cf	df
try	10422	8760
insurance	10440	3997

Tabelul 4.1: Frecvența la nivel de colecție (cf) și frecvența documentelor (df) se comportă diferit - exemplu din colecția *Reuters*

În loc de acest factor, se folosește de obicei frecvența documentelor df_t , definită ca numărul de documente care conțin termenul t . Tabelul 4.1 ilustrează motivul pentru care este preferat df . În mod intuitiv, vrem ca puținele documente care conțin *insurance* să aibă un scor mai mare față de restul pentru o interogare ce conține cuvântul *insurance*. Evident, același lucru nu este valabil și pentru *try*.

Notând cu N numărul de documente dintr-o colecție, definim frecvența inversă a documentelor pentru un termen t , idf_t astfel:

$$idf_t = \log \frac{N}{df_t} \quad (4.1)$$

Așadar idf-ul unui termen rar este ridicat, în timp ce idf-ul unui termen frecvent este scăzut. Tabelul 4.2 ilustrează valorile df și idf pentru câțiva termeni din colecția Reuters.

Termen	df_t	idf_t
car	18165	1.65
auto	6723	2.08
insurance	19241	1.62
best	25235	1.5

Tabelul 4.2: Exemplu de valori idf din colecția Reuters de 806791 documente.

4.1.3 Ponderarea $tf-idf$

Combinând definițiile celor două măsuri tf și idf , se obține o pondere compusă pentru fiecare termen din fiecare document. Schema de ponderare $tf-idf$ asignează termenului t în documentul d dată de:

$$tf-idf_{t,d} = tf_{t,d} \times idf_t. \quad (4.2)$$

Cu alte cuvinte, $tf-idf_{t,d}$ conferă termenului t din documentul d o pondere care:

1. are valorile cele mai mari când t are un număr mare de ocurențe în puține documente
2. este mai mică când nu apare de foarte multe ori în d sau apare în multe documente
3. are valoarea cea mai mică (zero) când apare în toate documentele.

În acest moment fiecare document poate fi vizualizat ca un vector cu câte o componentă pentru fiecare termen din dicționarul indexului, având valoarea dată de ecuația 4.2. Această formă este crucială în ierarhizarea în RI. O primă variantă de a calcula scorul unui document în raport cu o interogare folosind cele menționate până acum este:

$$Score(q, d) = \sum_{t \in q} tf-idf_{t,d} \quad (4.3)$$

4.1.4 Similaritatea cosinus

Reprezentarea unui set de documente ca vectori într-un spațiu vectorial este cunoscută sub numele de *modelul spațiului vectorial* (*VSM - vector space model*) și este vitală pentru o serie de operații precum obținerea scorului pentru o interogare, clasificare de documente și grupare (*clustering*). Un aspect foarte important al acestui model este reprezentarea interogării în același spațiu vectorial.

Produsul scalar

Notăm $\vec{V}(d)$ vectorul derivat din documentul d , cu câte o componentă pentru fiecare termen din dicționar. Presupunem că valoarea fiecărei componente este calculată folosind schema de ponderare $tf-idf$, deși schema de ponderare este nesemnificativă pentru model. Setul de documente din colecție poate fi privit acum ca un set de vectori într-un spațiu vectorial cu câte o axă pentru fiecare termen. Așa cum am mai menționat, acest tip de reprezentare ignoră ordinea termenilor (*bag of words*).

Se pune problema cuantificării similarității între două documente în acest spațiu vectorial. O primă idee ar fi ca aceasta să fie dată de vectorul diferență dintre vectorii celor două documente. Problema acestei abordări este că două documente cu conținut similar pot avea o distanță foarte mare între ele doar pentru că unul este mult mai lung decât celălalt, distribuția relativă a termenilor fiind identică (dar frecvențele absolute diferind).

Pentru a compensa pentru efectul lungimii documentelor, modul standard de a cuantifica similaritatea dintre două documente d_1 și d_2 este calcularea *similarității cosinus* ale celor două reprezentări vectoriale $\vec{V}(d_1)$ și $\vec{V}(d_2)$

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| |\vec{V}(d_2)|}, \quad (4.4)$$

unde numărătorul reprezintă produsul scalar, în timp ce numitorul este produsul lungimilor celor doi vectori. Produsul scalar a doi vectori $\vec{x} \cdot \vec{y}$ este definit ca $\sum_{i=1}^M x_i y_i$. Fie $\vec{V}(d)$ reprezentarea vectorială a documentului d cu M componente $V_1(d), \dots, V_M(d)$. Lungimea sau norma euclidiană este definită ca $\sqrt{\sum_{i=1}^M V_i^2(d)}$.

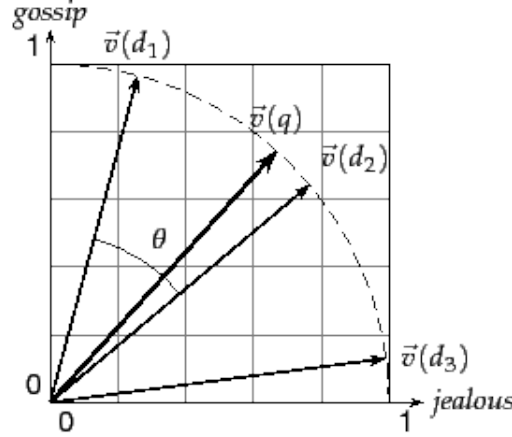


Figura 4.1: Similaritatea cosinus ilustrată pentru interogarea "jealous gossip". $\text{sim}(d_1, d_2) = \cos \theta$

Efectul numărătorilor este, așadar, să normalizeze vectorii $\vec{V}(d_1)$ și $\vec{V}(d_2)$ la vectorii unitate $\vec{v}(d_1) = \vec{V}(d_1)/|\vec{V}(d_1)|$ și $\vec{v}(d_2) = \vec{V}(d_2)/|\vec{V}(d_2)|$. Putem rescrie ecuația 4.4 ca:

$$\text{sim}(d_1, d_2) = \vec{v}(d_1) \cdot \vec{v}(d_2). \quad (4.5)$$

Ecuația 4.5 poate fi văzută ca produsul scalar dintre versiunile normalizate ale vectorilor celor două documente. Această valoare reprezintă cosinusul unghiului θ dintre cei doi vectori (vezi figura 4.1). Dat fiind un document d , măsura de similaritate poate fi folosită la găsirea altor documente asemănătoare lui d . Acest lucru poate fi dorit de un utilizator care identifică un document relevant la nevoia lui de informație și, pornind de la acest document, dorește să găsească alte documente relevante. O astfel de abordare poartă numele de *mai multe ca acesta - more like this*. Problema găsirii documentului d_i cel mai asemănător cu d se reduce la găsirea unui document d_i cu proprietatea că produsul scalar $\vec{v}(d) \cdot \vec{v}(d_i)$ este maxim.

Vizualizarea unei colecții de N documente ca o colecție de vectori conduce la vizualizarea colecției ca o *matrice termen-document*. Aceasta este o matrice de dimensiune $M \times N$ ale cărei rânduri reprezintă cei M termeni (dimensiuni) ale celor N coloane, fiecare corespunzând unui document.

Interogările văzute ca vectori

Există un motiv și mai convingător pentru reprezentarea documentelor ca vectori: reprezentarea interogărilor ca vectori. Ideea acum este să se asigneze fiecărui document d din colecție un scor egal cu produsul scalar $\vec{q} \cdot \vec{d}$.

Prin vizualizarea unei interogări ca "un sac de cuvinte" (*bag of words*), se poate trata ca un document foarte scurt. Scorurile rezultate în urma calculării similarităților fiecărui document cu interogarea pot fi apoi folosite pentru a selecta și ierarhiza documentele relevante.

$$\text{sim}(q, d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)| |\vec{V}(d)|}, \quad (4.6)$$

Un document poate avea un scor cosinus mare pentru o interogare chiar dacă nu conține toți termenii din interogare. Formula similarității cosinus nu se bazează pe o anumită ponderare a termenilor, schema de ponderare putând fi tf, tf-idf sau alte tipuri sau variațiuni.

Calcularea similarității cosinus între vectorul interogării și vectorii fiecărui document din colecție, sortarea sorurilor rezultate și selectarea primelor K documente pot fi operațiuni costisitoare - computarea unei singure similarități poate duce la calcularea unui produs scalar pentru mii de dimensiuni (numărul termenilor din colecție). De aceea există o serie de euristici folosite pentru a îmbunătăți timpul necesar efectuării calcului (vezi secțiunea 4.2).

Calcularea scorurilor

Într-o situație uzuală avem o colecție de documente, fiecare reprezentat de un vector, o interogare reprezentată de un vector și un întreg pozitiv K . Căutăm primele K documente din colecție cu sorurile date de similaritatea cosinus cele mai mari în contextul interogării date. De exemplu, multe motoare de căutare folosesc $K = 10$ pentru a regăsi și ierarhiza documentele afișate pe prima pagină (cele mai bune). Algoritmul 1 prezintă algoritmul de bază pentru computarea scorurilor.

Algoritmul 1 Calcularea similarității cosinus

```

1: procedure COSINESCORE( $q$ )                                ▷ primele  $K$  documente în ordinea similitudinii cosinus
2:   float  $Scores[N] \leftarrow 0$ 
3:   initialize  $Length[N]$ 
4:   for all  $t \in q$  do
5:     calculate  $w_{t,q}$                                        ▷ ponderea termenului  $t$  în interogarea  $q$ 
6:     fetch postings list  $P_t$  and  $idf_t$  for term  $t$ 
7:     for all  $(d, tf_{t,d}) \in P_t$  do
8:        $wf_{t,d} \leftarrow tf_{t,d} \times idf_t$                 ▷ sau orice altă schemă de ponderare
9:        $Score[d] \leftarrow Score[d] + wf_{t,d} \times w_{t,q}$     ▷ produsul scalar
10:    end for
11:  end for
12:  read array  $Length[d]$                                      ▷ norma euclidiană pentru fiecare document
13:  for all  $d$  do
14:     $Scores[d] \leftarrow Scores[d] / Length[d]$               ▷ normalizare
15:  end for
16:  return Top  $K$  components of  $Scores[]$ 
17: end procedure

```

Lista $Length$ ține lungimile vectorilor (factorii de normalizare) pentru fiecare dintre cele N documente, în timp ce lista $Scores$ ține scorul fiecărui document. După ce scorul este calculat, tot ceea ce rămâne de făcut este să se aleagă primele K documente cu scorul cel mai mare.

Prima buclă de la pasul 4 iterează peste fiecare termen din interogare actualizând scorul pentru fiecare document. La pasul 5 se calculează ponderea termenului t în vectorul interogării. Iterația de la pasul 7 actualizează fiecare document din lista de postare a termenului t cu contribuția aferentă. Acest proces este cunoscut sub numele de *acumulare* iar elementele listei $Scores$ sunt *acumulatori*. Pentru a evita stocarea numerelor în virgulă mobilă în index, se poate stoca df_t pentru fiecare termen, și $tf_{t,d}$ pentru fiecare document dintr-o listă de postare. La căutare, pentru fiecare termen t , se poate calcula idf_t și apoi ponderea tf-idf pentru fiecare document din lista de postare. La pasul 16 sunt extrase primele K documente; acest proces necesită o coadă cu prioritate implementată de obicei folosind ca structură de date un heap.

Algoritmul 1 nu prescrie o implementare specifică despre cum trebuie traversate listele de postare ale termenilor din interogare. Acestea pot fi traversate pentru fiecare termen în parte sau în mod concurent, pentru toți termenii, calculând la fiecare pas întregul scor al unui document (cu condiția ca documentele să fie ordonate după același criteriu în fiecare listă). Prima variantă poartă numele de *term-at-a-time* iar cea de-a doua, *document-at-a-time*.

4.1.5 Variante de ponderare tf-idf

Au fost propuse și alte alternative la tf și idf pentru ponderarea termenilor din documente.

Scalarea subliniară tf

Pare improbabil ca 20 de ocurențe ale unui termen într-un document au o însemnătate de 20 de ori mai mare decât o singură ocurență. Ca urmare, s-au căutat variante de luare în considerare a frecvenței termenilor folosind funcții care nu cresc liniar cu numărul de ocurențe. O abordare comună este folosirea logaritmului frecvenței termenilor:

$$wf_{t,d} = \begin{cases} 1 + \log tf_{t,d} & \text{dacă } tf_{t,d} > 0 \\ 0 & \text{altfel} \end{cases} \quad (4.7)$$

Utilizând această pondere în loc de tf , putem înlocui tf -idf cu:

$$wf-idf_{t,d} = wf_{t,d} \times idf_t. \quad (4.8)$$

Normalizarea cu tf maxim

O altă tehnică este normalizarea ponderilor tf pentru fiecare termen din document cu tf -ul maxim din acel document. Fie $tf_{max}(d) = \max_{\tau \in d} tf_{\tau,d}$ pentru fiecare document d . Frecvența normalizată a termenilor este calculată astfel:

$$ntf_{t,d} = a + (1 - a) \frac{tf_{t,d}}{tf_{max}(d)}, \quad (4.9)$$

unde a este o valoare între 0 și 1 (de obicei ia valoarea 0.4) și se numește *factor de uniformizare*. Are rolul de a amortiza contribuția fracției din ecuația 4.9 (ideea de bază este să se evite o oscilație mare a $ntf_{t,d}$ cauzată de o oscilație mică a $tf_{t,d}$).

Ideea principală din spatele normalizării cu tf maxim este evitarea următoarei anomalii: se observă frecvențe mai mari ale termenilor în documente mai lungi, în principal din cauza faptului că documentele mai lungi tind să repete aceleași cuvinte de mai multe ori, lucru care nu ar trebui să afecteze calcularea relevanței.

Scheme de ponderare pentru documente și interogări

Ecuația 4.6 este fundamentală pentru sisteme de regăsirea informației care folosesc orice formă de ierarhizare pe baza VSM. Diferențele dintre metodele VSM țin mai degrabă de alegerea ponderilor din vectorii $\vec{V}(d)$ și $\vec{V}(q)$. Tabelul 4.3 ilustrează unele dintre principalele scheme de ponderare folosite atât pentru $\vec{V}(d)$ cât și pentru $\vec{V}(q)$, împreună cu mnemonice pentru a reprezenta o combinație specifică de ponderi. Acest sistem de mnemonice poartă numele de *notația SMART*.

Frecvența termenilor(tf)	Frecvența în documente	Normalizare
n $tf_{t,d}$		n 1
l $1 + \log tf_{t,d}$		c $\frac{1}{\sqrt{\sum_{i=1}^M w_i^2}}$
a $0.5 + \frac{0.5 \times tf_{t,d}}{\max_t tf_{t,d}}$	n 1	
b $\begin{cases} 1 & \text{dacă } tf_{t,d} > 0 \\ 0 & \text{altfel} \end{cases}$	t $\log \frac{N}{df_t}$	u $1/u$
L $\frac{1 + \log tf_{t,d}}{1 + \log \max_{t \in d} (tf_{t,d})}$	p $\max \left\{ 0, \log \frac{N - df_t}{df_t} \right\}$	b $1/Cl^\alpha, \alpha < 1$

Tabelul 4.3: Notația *SMART* pentru variantele tf -idf; Cl reprezintă numărul de caractere din document

Notația pentru reprezentarea unei combinații de scheme de ponderare are forma $ddd.qqq$, unde primul triplet reprezintă ponderarea pentru vectorul documentului iar cel de-al doilea pentru interogare. Prima literă din fiecare triplet reprezintă componenta frecvenței termenului, cea de-a doua, frecvența în documente, iar cea de-a treia, forma de normalizare folosită. În practică, se aplică scheme diferite de normalizare pentru document, respectiv interogare. De exemplu, o schemă de ponderare des întâlnită este *lnc.ltc*.

4.2 Euristici pentru eficientizare

Revizualizând algoritmul 1 pentru o interogare $q = \textit{jealous gossip}$, două observații sunt imediate:

1. Vectorul $\vec{v}(q)$ are două componente diferite de zero.
2. În absența unei ponderări pentru termenii interogării, cele două componente nenule au aceeași valoare.

Pentru a ierarhiza documentele relevante la interogare, suntem interesați de scorurile relative ale documentelor din colecție. Așadar, este suficientă calcularea similarității cosinus între fiecare vector normalizat $\vec{v}(d)$ și $\vec{V}(q)$ (în care toate componentele nenule sunt egale cu 1) în loc de $\vec{v}(q)$. Pentru oricare două documente d_1, d_2

$$\vec{V}(q) \cdot \vec{v}(d_1) > \vec{V}(q) \cdot \vec{v}(d_2) \Leftrightarrow \vec{v}(q) \cdot \vec{v}(d_1) > \vec{v}(q) \cdot \vec{v}(d_2). \quad (4.10)$$

Pentru fiecare document d , similaritatea cosinus $\vec{V}(q) \cdot \vec{v}(d)$ este suma peste toți termenii din interogarea q a ponderilor acestor termeni în d . Această sumă se poate calcula prin intersectarea listelor de postare ca în algoritmul 1, cu mențiunea că ponderile interogării sunt considerate 1. Această schemă calculează un scor pentru fiecare document din listele de postare ale termenilor din interogare (numărul de documente poate fi considerabil mai mic decât N).

Algoritmul 2 Un algoritm mai rapid pentru calcularea similarității cosinus

```

1: procedure COSINESCORE( $q$ )                                ▷ primele K documente în ordinea similității cosinus
2:   float  $Scores[N] \leftarrow 0$ 
3:   initialize  $Length[N]$ 
4:   for all  $t \in q$  do
5:     fetch postings list  $P_t$  and  $idf_t$  for term  $t$ 
6:     for all  $(d, tf_{t,d}) \in P_t$  do
7:        $wf_{t,d} \leftarrow tf_{t,d} \times idf_t$                                 ▷ sau orice altă schemă de ponderare
8:        $Score[d] \leftarrow Score[d] + wf_{t,d}$ 
9:     end for
10:  end for
11:  read array  $Length[d]$                                 ▷ norma euclidiană pentru fiecare document
12:  for all  $d$  do
13:     $Scores[d] \leftarrow Score[d] / Length[d]$                                 ▷ normalizare
14:  end for
15:  return Top K components of  $Scores[]$ 
16: end procedure

```

După calcularea scorurilor, pasul final înainte de prezentarea rezultatelor unui utilizator este să se aleagă K documente cu cele mai bune scoruri. Se poate folosi o sortare dar o abordare mai bună este folosirea unui *heap* pentru a păstra ordinea documentelor. Dacă J este numărul de documente cu scoruri nenule, construirea acestui heap poate fi făcută în $2J$ pași.

4.2.1 Regăsirea inexactă a primelor K documente

Până acum scopul primordial a fost regăsirea *precisă* a primelor K documente pentru o interogare. În continuare voi prezenta scheme cu ajutorul cărora sunt produse K documente pentru care șansele sunt foarte mari să facă parte din *primele* K . Procedând astfel, scopul este de a micșora semnificativ costul regăsirii, fără a altera percepția utilizatorului asupra relevanței documentelor întoarse. În secțiunile următoare sunt detaliate euristici care realizează acest lucru.

O astfel de regăsire inexactă nu e în mod necesar, din perspectiva utilizatorului, un lucru rău. Documentele din *top-K* după similaritatea cosinus nu sunt neapărat cele mai bune K pentru nevoia respectivă de informație. Principalul cost decurge din calcularea similarității cosinus între interogare și un număr mare de documente. Un număr mare de documente duce și la creșterea costului de selecție în faza finală de extragere din heap.

Vom considera, în continuare o serie de idei care au ca scop eliminarea unui număr mare de documente fără a mai calcula similaritatea. Euristicile au la bază următorii pași:

1. Găsește un set A de documente pretendente, unde $K < |A| \ll N$. A nu conține neapărat cele mai bune K documente, dar șansele sunt foarte mari să conțină o bună parte din acestea.
2. Întoarce cele mai bune K documente din A .

Din descrierea acestor idei, este clar că multe dintre euristici vor necesita parametri care să fie ajustați la colecție și aplicație. Majoritatea acestor euristici sunt adecvate pentru interogări de tip "free text" dar nu și pentru interogări booleane sau "frază exactă".

4.2.2 Ignorarea termenilor

Pentru o interogare multi-termen q , este evident că se vor lua în calcul doar documente care conțin cel puțin unul din termeni. Acest lucru poate fi extins folosind euristici adiționale:

1. Se consideră doar documente care conțin termeni ale căror ponderi *idf* sunt mai mari decât o valoare prestabilită. Așadar, la traversarea listelor de postare, se traversează doar listele termenilor cu *idf* mare. Acest lucru aduce un beneficiu mare: listele de postare ale termenilor cu ponderi *idf* mici sunt în general lungi; prin îndepărtarea lor setul de documente pentru care trebuie calculată similaritatea scade considerabil. O altă perspectivă asupra acestei euristici este că termenii cu *idf* redus sunt considerați cuvinte de stop și nu contribuie la calcularea scorului.
2. Se consideră doar documente care conțin mulți (sau toți) termeni din interogare. Acest lucru poate fi realizat la traversarea postărilor. Un pericol al acestei scheme este că dacă se elimină prea multe documente setul de documente A va avea mai puțin de K documente.

Aceste metode euristice intră în categoria *index elimination*.

4.2.3 Listele de campioni

Ideea din spatele *listelor de campioni* (*champion lists*, *fancy lists*, *top docs*) este precomputarea, pentru fiecare termen t din dicționar, unui set de r documente care au ponderi foarte mari pentru t ; valoare lui r este aleasă inițial. Pentru ponderarea *tf-idf*, acestea ar fi primele r cu ponderea *tf* cea mai mare pentru termenul t . Numim această mulțime de r documente *lista de campioni* a termenului t .

Având o interogare q , creăm setul A astfel: luăm reuniunea listelor de campioni ale fiecărui termen din q . Restricționăm calcularea similarității cosinus doar la documentele din A . Un parametru critic al acestei scheme este valoarea lui r , care variază în funcție de aplicație. Intuitiv, r ar trebui să fie mult mai mare decât K , mai ales dacă se folosesc euristicele prezentate în secțiunea precedentă. O problemă în cazul acesta este că r este setată la construirea indexului, în timp ce K este dependent de aplicație și există posibilitatea să fie aflat doar la lansarea interogării. Ca urmare, se poate întâmpla ca setul A să aibă mai puțin de K documente. Valoarea lui r ar putea fi setată mai mare pentru termeni mai rari.

4.2.4 Scoruri statice

Extindem listele de campioni la noțiunea de *scoruri statice*. În multe motoare de căutare pentru fiecare document d este disponibilă o măsură a calității $g(d)$ independentă de interogare, deci statică. Această măsură poate fi privită ca o valoare între 0 și 1. De exemplu, în contextul unei știri de pe web, $g(d)$ ar putea fi derivată din numărul de review-uri favorabile primite de la vizitatori.

Scorul net pentru documentul d este o combinație între $g(d)$ și scorul calculat la momentul interogării după o anumită schemă. Combinația poate fi determinată printr-un proces de învățare, dar, de dragul simplității, vom considera următoarea formă:

$$score(q, d) = g(d) + \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)| |\vec{V}(d)|}. \quad (4.11)$$

În această formă simplăm scorul static $g(d)$, și scorul calculat la interogare au contribuții egale, presupunând că ambele se află între 0 și 1.

Mai întâi să considerăm ordonarea documentelor din lista de postare a fiecărui termen în mod descrescător după valoarea lui $g(d)$. Acest lucru permite efectuarea intersecției listelor de postare în mod

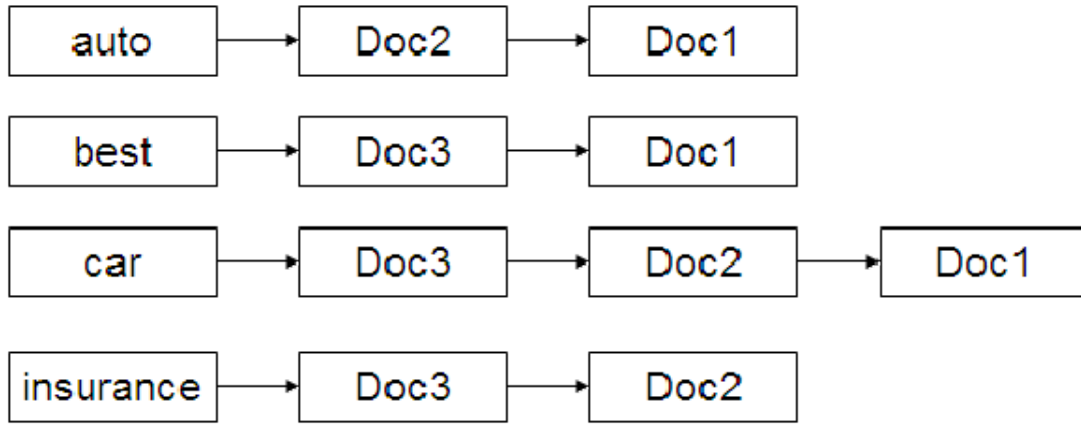


Figura 4.2: Index ordonat static. $g(1) = 0.25$, $g(2) = 0.5$, $g(3) = 1$

concurrent ca și cum ar fi fost ordonate după id-ul documentului (e nevoie doar de o ordonare comună pentru toate listele de postare). Acest tip de ordonare este ilustrat în figura 4.2.

Una dintre idei este extinderea directă a listelor de campioni: pentru o valoare bine aleasă r , menținem pentru fiecare termen t , o listă de r documente cu valorile cele mai mari ale $g(d) + tdf_{t,d}$. Lista este sortată după o ordonare comună. Apoi, la interogare, se calculează scorul net doar pentru documentele din reuniunea acestor liste de campioni.

Cea de-a doua idee constă în menținerea, pentru fiecare termen t , a două liste de postare disjuncte, fiecare sortată după $g(d)$. Prima listă, numită *high*, conține cele m documente cu valorile tf cele mai mari pentru t . Cea de-a doua listă numită *low*, conține toate celelalte documente ce îl conțin pe t . Când se procesează o interogare, se scanează mai întâi lista *high*, calculând scorurile nete pentru fiecare document (sau documente care conțin un număr mare din termenii interogării). Dacă nu se obțin K documente în urma acestui proces, se continuă cu listele *low*.

4.2.5 Impact ordering

În toate listele de postare descrise până acum, documentele au fost ordonate după o ordonare comună (după id-ul documentului sau scoruri statice). O astfel de ordonare comună suportă traversarea concurrentă a listelor de postare, calculând scorul pentru fiecare document în momentul în care acesta este întâlnit. O astfel de abordare este numită notare *document-at-a-time*. În această secțiune voi vorbi despre o tehnică în care listele de postare nu sunt toate ordonate după o ordonare comună, lucru care împiedică o traversare concurrentă. Ca urmare, scorurile vor fi "acumulate" luând termenii pe rând ca în algoritmul 1 (*term-at-a-time*).

Idee constă în ordonarea documentelor din listele de postare descrescător după $tf_{t,d}$. Ca urmare, ordonarea va fi diferită de la o listă de postări la alta și nu se poate face o traversare concurrentă pentru toți termenii din interogare. Având listele ordonate descrescător după $tf_{t,d}$, două idei s-au remarcat în a reduce substanțial numărul de documente pentru care trebuie acumulate scorurile:

1. când se traversează lista unui termen t , algoritmul se oprește după ce parcurge un anumit prefix al acesteia - fie după un număr fix r de intrări, fie după ce $tf_{t,d}$ a scăzut sub o anumită valoare presetată;
2. termenii interogării se parcurg în ordinea descrescătoare a idf , astfel încât termenii care au o probabilitate mai mare să contribuie mai mult la scorurile finale să fie luați în considerare primii.

Această ultimă idee poate fi adaptată la momentul procesării interogării: când se ajunge la termenii ai interogării cu idf scăzut, se poate determina dacă se continuă pe baza schibărilor scorurilor documentelor de la parcurgerea listei termenului anterior. Dacă schimbările sunt minime, se poate omite procesarea listei, sau se poate procesa doar un prefix.

Aceste idei vin dintr-o generealizare a metodelor introduse în secțiunile precedente. Depinzând de metoda de ponderare, listele de postare pot fi ordonate după alte cantități decât frecvența termenilor. În contextul RI această metodă poartă numele de *impact ordering*.

4.2.6 Cluster pruning

În *cluster pruning* există o etapă de preprocesare în cadrul căreia vectorii documentelor sunt grupați. Apoi, la momentul interogării, se consideră doar documentele dintr-un număr mic de grupuri drept candidați pentru care se calculează similaritatea. Pașii de preprocesare sunt următorii:

1. Alege \sqrt{N} documente din colecție în mod aleator. Numește aceste documente *lideri*.
2. Pentru fiecare document care nu este lider, află care este cel mai "apropiat" lider.

Documentele care nu sunt lideri vor fi numite *adepti*. Intuitiv, numărul aproximativ de *adepti* pentru fiecare lider este $\approx N/\sqrt{N} = \sqrt{N}$. Procesul de interogare decurge în felul următor:

1. Dacă fiind o interogare q , găsește liderul L cel mai apropiat de q . Acest lucru presupune calcularea similarităților cosinus între q și fiecare dintre cei \sqrt{N} lideri.
2. Setul de candidați A este alcătuit din L și adepii acestuia. Se calculează scorurile pentru toate documentele din acest set.

Folosirea liderilor aleși aleator pentru grupare este rapidă și probabilitatea să reflecte distribuția vectorilor în spațiu este destul de mare. Acest lucru este ilustrat în figura 4.3.

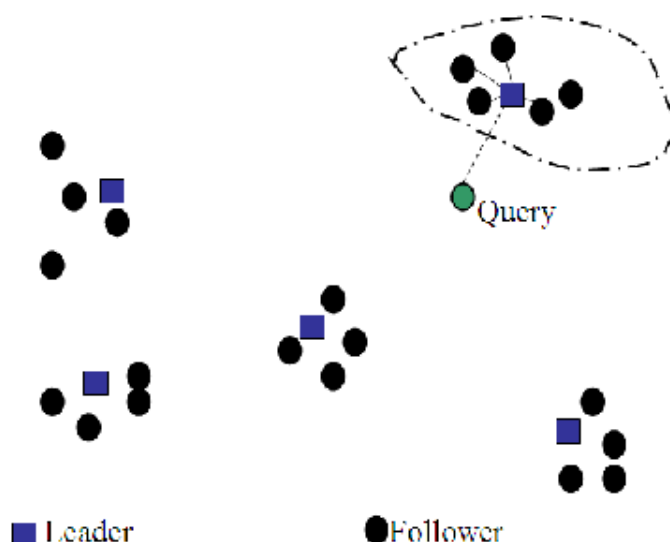


Figura 4.3: Cluster pruning

Variații ale acestei metode introduc parametri adiționali b_1 și b_2 , întregi pozitivi. În faza de preprocesare se atașează fiecare adept la cei mai apropiați b_1 lideri. La procesul de căutare se consideră cei mai apropiați b_2 lideri de interogarea q . Schema de mai sus corespunde cazului $b_1 = b_2 = 1$. Creșterea celor doi parametri duce la creșterea șanselor de a găsi K documente care să facă parte din *top-K*, dar necesită mai multe computații.

4.3 Modelul probabilistic

Dacă s-ar cunoaște relevanța unui subset de documente, s-ar putea estima probabilitatea apariției unui termen t într-un document relevant $P(t|R = 1)$ și, ca urmare, acesta ar putea reprezenta baza unui clasificator care decide dacă un document este relevant sau nu.

Utilizatorii încep cu *nevoi de informație* pe care le transformă în *interogări*. În mod similar, documentele sunt transformate în *reprezentări de documente* care diferă de primele cel puțin prin felul în care textul este împărțit în token-i. Bazându-se pe aceste două reprezentări, un sistem încearcă să determine cât de bine satisfac documentele nevoile de informații. În modelul Boolean sau VSM, dându-se numai o interogare, pentru un sistem RI nevoia de informație este incertă. Dându-se interogarea și reprezentarea

documentelor, un sistem trebuie să "ghicească" dacă un document are conținut relevant pentru respectiva nevoie de informație. Teoria probabilităților pune la dispoziție o fundație de principii potrivite pentru raționament în situații incerte. Aceste principii pot fi exploatate pentru a estima cât de probabil este ca un document să fie relevant pentru o nevoie de informație.

Există mai multe posibile modele probabilistice de regăsire. În continuare voi discuta despre *principiul probabilistic de ierarhizare* și despre *modelul binar de independență*, care a fost primul model probabilistic de regăsire. În final voi prezenta și sistemul de ponderare *Okapi BM25*, care a avut un succes destul de mare în practică.

În acest context, este util conceptul de șanse (*odds*), pe lângă cel de probabilitate.

$$Odds : O(A) = \frac{P(A)}{P(\bar{A})} = \frac{P(A)}{1 - P(A)}. \quad (4.12)$$

4.3.1 Principiul probabilistic de ierarhizare

Cazul 1/0 loss

Presupunem că sistemul de RI întoarce ca răspuns la o interogare o listă ordonată de documente și folosirea unei notații binare pentru relevanță. Pentru o interogare q și un document d , fie $R_{d,q}$ o variabilă aleatoare care indică dacă d este relevant în contextul interogării q . Variabila ia valoarea 1 când documentul este relevant și 0 altfel.

Folosind un model probabilistic, ordinea evidentă în care documentele trebuie prezentate utilizatorului este dată de ierarhizarea documentelor după probabilitatea estimată de relevanță în raport cu nevoia de informație: $P(R_{d,q} = 1|d, q)$. Vom scrie R în loc de $R_{d,q}$. Aceasta reprezintă temelia *principiului probabilistic de ierarhizare (PRP)*:

Dacă răspunsul unui sistem la fiecare cerere este o ierarhizare a documentelor din colecție în ordinea descrescătoare a probabilității de relevanță, unde probabilitățile sunt estimate cât mai bine cu putință pe baza datelor pe care sistemul le are la îndemână, eficiența sistemului este cea mai bună care se poate obține folosind aceste date.

În cel mai simplu caz al PRP, nu există costuri de regăsire sau alte motive de îngrijorare care să valorifice diferit acțiunile sau erorile. Se pierde un punct fie pentru întoarcerea unui document nerelevant, fie pentru lipsa întoarcerii unui document relevant. O astfel de evaluare binară asupra preciziei poartă numele de *1/0 loss*. Scopul este să se întoarcă cele mai bune k rezultate posibile, pentru orice valoare k aleasă de utilizator. PRP spune că documentele trebuie ierarhizate în ordinea descrescătoare a $P(R = 1|d, q)$.

Teoremă 1 *PRP este optim în sensul că minimizează pierderea așteptată (sau riscul Bayes) în cazul 1/0 loss.*

Această teoremă este adevărată dacă toate probabilitățile sunt corecte, ceea ce în practică este imposibil. Cu toate acestea, PRP reprezintă o fundație pentru contruirea de modele de RI.

Costuri de regăsire

Să presupunem existența unui model de costuri de regăsire. Fie C_1 costul de regăsire a unui document relevant și C_0 costul de regăsire a unui document nerelevant. Atunci pentru un document d și pentru toate documentele d' neregăsite dacă

$$C_1 \times P(R = 1|d) + C_0 \times P(R = 0|d) \leq C_1 \times P(R = 1|d') + C_0 \times P(R = 0|d') \quad (4.13)$$

atunci d este următorul document care trebuie întors. Acest model asigură un cadru formal în care putem modela costurile diferențiale ale falselor-pozitive și falselor-negative.

4.3.2 Modelul binar de independență

BIM este modelul care a fost folosit cu PRP. Introduce câteva asumții simple care permit estimarea funcției probabilistice $P(R|d, q)$. Aici binar este echivalent cu boolean: atât documentele cât și interogările sunt reprezentate ca vectori binari de incidență a termenilor. Un document d este reprezentat de vectorul $\vec{x} = (x_1, \dots, x_M)$, unde $x_t = 1$ dacă termenul t este prezent în documentul d și $x_t = 0$ altfel. În contextul acestei reprezentări, multe documente pot avea aceeași reprezentare. În mod similar, interogarea q este reprezentată prin vectorul de incidență \vec{q} . "Independență" se referă la faptul că termenii sunt modelați așa cum apar în documente în mod independent. Modelul nu recunoaște niciun tip de asociere între termeni. Această asumție este, evident, incorectă, dar, în pofida acestui aspect, oferă rezultate satisfăcătoare în practică. Este asumția care stă și la baza modelului *Bayes Naiv*, și este într-un fel echivalentă cu asumția din VSM, în care fiecare termen reprezintă o dimensiune ortogonală față de celelalte.

Pentru a face o strategie probabilistică de regăsire precisă, trebuie estimat modul în care termenii din documente contribuie la relevanță. Cu alte cuvinte, trebuie să specificăm cum frecvența termenilor într-un document, numărul de documente care conțin un termen, lungimea unui document și alte statistici influențează calculul relevanței unui document. După acest proces documentele vor fi ordonate în ordinea descrescătoare a acestor probabilități estimate.

Se pornește de la asumția că relevanța fiecărui document este independentă de relevanța celorlalte documente. În practică, acest lucru pune o problemă în momentul în care sunt întoarse documente duplicate sau aproape duplicate. În contextul BIM, probabilitatea că un document este relevant la o interogare $P(R|d, q)$ este modelată via probabilitatea $P(R|\vec{x}, \vec{q})$, folosind vectorii de incidență. Apoi, aplicând regula lui Bayes, se obține:

$$P(R = 1|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}, \vec{q})} \quad P(R = 0|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}, \vec{q})} \quad (4.14)$$

Aici, $P(\vec{x}|R = 1, \vec{q})$ și $P(\vec{x}|R = 0, \vec{q})$ reprezintă probabilitatea ca dacă un document relevant, respectiv nerelevant, este întors, acesta să aibă reprezentarea \vec{x} . Aceste probabilități nu se pot calcula exact, așa că trebuie folosiți estimatori: statistici ale colecției de documente sunt folosite pentru a estima aceste probabilități. $P(R = 1|\vec{q})$ și $P(R = 0|\vec{q})$ reprezintă probabilitatea apriori de a întoarce un document relevant, respectiv nerelevant, dată fiind interogarea \vec{q} . Pentru că un document este fie relevant fie nerelevant în contextul unei interogări, avem:

$$P(R = 1|\vec{x}, \vec{q}) + P(R = 0|\vec{x}, \vec{q}) = 1. \quad (4.15)$$

Derivarea unei funcții de ierarhizare

Decât să estimăm $P(R = 1|\vec{x}, \vec{q})$ direct, deoarece interesează doar ordinea în care sunt întoarse documentele, folosim alte cantități care sunt mai ușor de calculat și care au ca rezultat aceeași ordine. Putem ordona documentele după șansele (odds) de relevanță, ceea ce duce la o simplificare a relației:

$$O(R|\vec{x}, \vec{q}) = \frac{P(R = 1|\vec{x}, \vec{q})}{P(R = 0|\vec{x}, \vec{q})} = \frac{\frac{P(\vec{x}|R=1, \vec{q})P(R=1|\vec{q})}{P(\vec{x}, \vec{q})}}{\frac{P(\vec{x}|R=0, \vec{q})P(R=0|\vec{q})}{P(\vec{x}, \vec{q})}} = \frac{P(R = 1|\vec{q})}{P(R = 0|\vec{q})} \cdot \frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})} \quad (4.16)$$

Termenul stâng al expresiei din dreapta al ecuației 4.16 este constant pentru o interogare dată. Pentru că interesează doar ordinea, nu e nevoie să se estimeze. Trebuie estimat, în schimb, celălalt termen, lucru care pare dificil inițial: cum poate fie precis estimată probabilitatea unui întreg vector de incidență? Pentru a face posibilă estimarea, se face asumția de *condițional-independență Naive Bayes*: prezența sau absența unui cuvânt într-un document este independentă de prezența sau absența oricărui alt cuvânt:

$$\frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})} = \prod_{t=1}^M \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})} \quad (4.17)$$

Așadar:

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^M \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})}. \quad (4.18)$$

Pentru că fiecare x_t este fie 0, fie 1, putem separa termenii astfel:

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t=1|R=1, \vec{q})}{P(x_t=1|R=0, \vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1, \vec{q})}{P(x_t=0|R=0, \vec{q})}. \quad (4.19)$$

Fie $p_t = P(x_t=1|R=1, \vec{q})$ probabilitatea ca termenul x_t să apară într-un document relevant la interogarea dată și $u_t = P(x_t=1|R=0, \vec{q})$, probabilitatea ca termenul să apară într-un document nerelevant. Aceste cantități pot fi vizualizate în tabelul de contingență următor (suma pe coloane este 1):

	document	relevant (R=1)	nerelevant(R=0)
termen present	$x_t = 1$	p_t	u_t
termen absent	$x_t = 0$	$1 - p_t$	$1 - u_t$

Facând asumția că termenii care nu apar în interogare au probabilități egale de apariție într-un document relevant, respectiv nerelevant: $q_t = 0 \Rightarrow p_t = u_t$, vor trebui luați în considerare doar termenii care apar în interogare:

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0, q_t=1} \frac{1-p_t}{1-u_t}. \quad (4.20)$$

Primul produs este peste termenii interogării care apar în document și produsul din dreapta peste cei care nu apar.

Expresia poate fi manipulată prin includerea termenilor găsiți în document în produsul din dreapta, dar, în același timp, ajustând produsul stâng pentru simplificare:

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}. \quad (4.21)$$

Produsul drept este acum peste toți termenii interogării, ceea ce înseamnă că e constant pentru o interogare, la fel ca $O(R|\vec{q})$. Așa dar, singura cantitate care trebuie estimată pentru a ierarhiza documentele este cea din produsul stâng. Putem ordona documentele și după rezultatul logaritmului produsului, deoarece log este o funcție monotonă. Cantitatea folosită la ierarhizare se numește *valoarea statusului de regăsire* (*RSV - retrieval status value*):

$$RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}. \quad (4.22)$$

Totul se reduce la clacularea RSV. Definim c_t :

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{1-p_t} + \log \frac{1-u_t}{u_t}. \quad (4.23)$$

Termenii c_t reprezintă proporțiile *log odds* pentru termenii interogării. Valoare va fi 0 dacă un termen are șanse egale să apară într-un document relevant, respectiv nerelevant, și pozitivă dacă este mai probabil să apară într-un document relevant. Cantitățile c_t funcționează ca ponderi ale termenilor în model, iar scorul pentru o interogare și un document este: $RSV_d = \sum_{t:x_t=q_t=1} c_t$. Problema rămasă este cum să se estimeze cantitățile c_t pentru o colecție de documente și o interogare.

Estimări teoretice

Următorul tabel de contingență prezintă o serie de statistici ale colecției, unde df_t reprezintă numărul de documente care conțin termenul t :

Așadar, $p_t = s/S$ și $u_t = (df_t - s)/(N - S)$ și

$$c_t = K(N, df_t, S, s) = \log \frac{s/(S-s)}{(df_t - s)/((N - df_t) - (S - s))}. \quad (4.24)$$

	document	relevante	nerelevante	total
termen prezent	$x_t = 1$	s	$df_t - s$	df_t
termen absent	$x_t = 0$	$S - s$	$(N - df_t) - (S - s)$	$N - df_t$
	total	S	$N - S$	N

Pentru a evita posibilitatea apariției de zerouri (de exemplu, un anumit termen este conținut fie de toate, fie de niciun document relevant) se adaugă $\frac{1}{2}$ la fiecare dintre cele 4 cantități și apoi se ajustează totalurile $(N + 2)$. Ca urmare avem:

$$\hat{c}_t = K(N, df_t, S, s) = \log \frac{(s + \frac{1}{2}) / (S - s + \frac{1}{2})}{(df_t - s + \frac{1}{2}) / (N - df_t - S + s + \frac{1}{2})}. \quad (4.25)$$

Adăugarea valorii $\frac{1}{2}$ este o formă simplă de *uniformizare*.

Estimări practice

Sub asumția că documentele relevante reprezintă un procent infim din colecție, este plauzibilă aproximarea statisticilor pentru documentele nerelevante cu statisticile pe întreaga colecție. Ca urmare, u_t (probabilitatea ca un document nerelevant să conțină termenul t pentru o interogare) poate fi aproximat cu df_t/N și:

$$\log \frac{1 - u_t}{u_t} = \log \frac{N - df_t}{df_t} \approx \log \frac{N}{df_t} \quad (4.26)$$

Rezultatul este interesant și prin faptul că furnizează o justificare teoretică a celei mai întâlnite forme de ponderare *idf* folosită în VSM.

Tehnica de aproximare din ecuația 4.26 nu poate fi ușor extinsă la documente relevante. Cantitatea p_t poate fi estimată în mai multe moduri:

1. Se poate folosi frecvența termenilor din documentele cunoscute deja ca fiind relevante (dacă se cunosc - metodă folosită în cadrul *feedback-ului de relevanță*)
2. Se poate presupune că fiecare termen are șanse egale să apară într-un document relevant: $p_t = 0.5$. Această estimare este destul de slabă. Combinând această metodă cu aproximarea lui u_t de mai sus, ierarhizarea documentelor este dată de termenii interogării care apar în documente scalați cu ponderea *idf*.
3. O altă aproximare propusă folosește statisticile aparițiilor termenilor în colecție: $p_t = df_t/N$.

4.3.3 Okapi BM25

Metodele probabiliste sunt unele din cele mai vechi modele formale în RI. Încă din 1970 erau privite ca o oportunitate pentru a pune bazele teoretice în RI și, odată cu "renașterea" modelelor probabiliste în lingvistica computațională în anii 1990, această oportunitate s-a întors iar metodele probabiliste reprezintă unul dintre subiectele cele mai discutate subiecte în materie de RI. Obținerea unor aproximări rezonabile ale probabilităților necesare pentru un model RI probabilistic este posibilă, dar necesită prezumții majore. În modelul BIM acestea sunt:

- o reprezentare booleană a documentelor, interogărilor, relevanței
- independența termenilor
- termenii care nu apar în interogare nu afectează rezultatul
- valorile de relevanță ale documentelor sunt independente

Poate că din cauza severității asumpțiilor de modelare este dificilă obținerea unei performanțe mai bune. O problemă generală pare să fie că modelele probabiliste fie necesită informații parțiale de relevanță, fie duc la derivarea unor scheme aparent inferioare de ponderare a termenilor.

Această situație s-a schimbat în anii 1990 când schema de ponderare *BM25* a avut rezultate foarte bune și a început să fie adoptată de multe sisteme RI. Diferența dintre sistemele RI bazate pe *spatiul*

vectorial și cele bazate pe modelul probabilistic nu este așa de mare; în ambele cazuri se construiește un sistem similar, singura diferență fiind că scorul documentelor în contextul unei interogări este dat pe de-o parte de *similaritatea cosinus* aplicată pe vectori de ponderi *tf-idf*, iar pe de altă parte de o formulă ușor diferită motivată de teoria probabilităților.

Un model nebinar

Modelul BIM a fost inițial proiectat pentru scurte înregistrări de cataloage și a funcționat destul de bine în acest context, dar pentru căutări *full-text* pe colecții mari este evident că un model trebuie să ia în considerare frecvența termenilor și lungimea documentelor. Schema de ponderare numită Okapi după sistemul în care a fost inițial implementată, a fost proiectată folosind un model probabilistic sensibil la aceste tipuri de informație fără să introducă prea mulți parametri adiționali.

Cel mai simplu scor pentru un document d este dat de adunarea ponderilor *idf* ale termenilor în interogării prezente în document:

$$RSV_d = \sum_{t \in q} \log \frac{N}{df_t} \quad (4.27)$$

Pornind de la formula din ecuația 4.25, și estimând $S = s = 0$ în absența feedback-ului de relevanță, se obține o formulare alternativă a *idf*:

$$RSV_d = \sum_{t \in q} \log \frac{N - df_t + \frac{1}{2}}{df_t + \frac{1}{2}} \quad (4.28)$$

Această variantă are un comportament ciudat: dacă un termen apare în peste jumătate din documentele din colecție, modelul dă o pondere negativă, lucru care este nedorit. În cazul folosirii unui *stop list* acest lucru nu se întâmplă de obicei.

Ecuația 4.27 poate fi îmbunătățită prin folosirea frecvenței termenilor și a lungimii documentului:

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}} \quad (4.29)$$

Aici, tf_{td} este frecvența termenului t în documentul d și L_d și L_{ave} sunt lungimea documentului d , respectiv lungimea media a unui document din colecție. Variabila k_1 este un parametru pozitiv de ajustare care calibrează scalarea frecvenței tf_{td} : $k_1 = 0$ corespunde modelului binar, iar o valoare mare corespunde folosirii frecvenței brute. b este un alt parametru de calibrare ($0 \leq b \leq 1$) care determină scalarea după lungimea documentului: $b = 1$ presupune scalarea completă a ponderii termenului cu lungimea documentului în timp ce $b = 0$ presupune lipsa normalizării cu lungimea.

Dacă interogarea este lungă, atunci am putea folosi o ponderare similară pentru termenii din interogare. Acest lucru este adecvat dacă interogările au lungimi de dimensiunile unui paragraf, alfel este necesar:

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}} \cdot \frac{(k_3 + 1)tf_{tq}}{k_3 + tf_{tq}}. \quad (4.30)$$

Aici tf_{tq} este frecvența termenului t în interogarea și k_3 este un alt parametru pozitiv de calibrare care ajustează scalarea frecvenței termenilor din interogare. Parametru b pentru normalizarea lungimii interogării este necesar.

Acești parametri în mod ideal sunt setați să optimizeze performanța pe o colecție de test. Căutarea valorilor care să maximizeze performanța poate fi făcută manual sau automat. În absența unor astfel de optimizări, experimentele au arătat că valori bune pentru acești parametri sunt $b = 0.75$ și $1.2 \leq k_1, k_3 \leq 2$.

Formulele BM25 de ponderare a termenilor au fost folosite cu succes pe o varietate de colecții și tipuri de căutare. Au avut o performanță extrem de bună la evaluările TREC și au fost implementate în multe sisteme RI.

4.4 O abordare axiomatică

Este dificil, dacă nu imposibil, să se prezică performanța unui model de regăsire într-un mod analitic. În continuare voi prezenta o abordare *axiomatică* ce poate fi folosită la proiectarea de noi modele de

ierarhizare în regăsirea informației bazate pe modelarea directă a relevanței cu constrângeri de regăsire formalizate definite la nivelul termenilor. Ideea de bază a acestei abordări este căutare într-un spațiu de funcții candidat a unei funcții care satisface un set de constrângeri (*axiome*) de regăsire. Pentru a defini spațiul de funcții, se va defini o funcție de regăsire în mod inductiv și se va descompune în trei componente. În [7] s-a constatat că euristicele intuitive de regăsire pot fi formal definite ca axiome și că performanța empirică a unei funcții de ierarhizare este legată strâns de cât de bine sunt satisfăcute aceste constrângeri.

Pentru a defini un model axiomatic pentru RI, trebuie să se definească un *spațiu de căutare* ale potențialelor funcții și un set de *constrângeri de regăsire* pe care orice funcție rezonabilă trebuie să le satisfacă. Asumția este că dacă o funcție satisface toate constrângerile, atunci are toate șansele să aibă rezultate foarte bune în practică. Spațiul de căutare trebuie să fie destul de mare încât să cuprindă funcții "bune", și suficient de mic pentru o căutare.

4.4.1 Spațiul funcțiilor

Din moment ce o funcție de regăsire este definită pe un document și o interogare, trebuie, mai întâi, să se definească documentele și interogările. Păstrând perspectiva metodelor curente de ierarhizare, documentele și interogările vor fi văzute ca "bag of words".

Fie T mulțimea tuturor termenilor. Fie interogarea $Q = \{q_1, \dots, q_n\}$ și documentul $D = \{d_1, \dots, d_m\}$, două seturi de termeni cu $q_i, d_i \in T$. Este posibil ca $q_i = q_j$ sau $d_i = d_j$ chiar dacă $i \neq j$. Scopul este definirea unei funcții de scor $S(Q, D) \in \mathbb{R}$. Pentru facilitarea căutării în spațiul de funcții și definirii de constrângeri, funcția de căutare va fi construită inductiv.

Vom începe cu cazul de bază în care atât documentul cât și interogarea conțin un singur termen.

Caz de bază: Presupunem că $Q = q$ și $D = d$.

$$S(Q, D) = f(q, d) = \begin{cases} \text{weight}(q) = \text{weight}(d), & q = d \\ \text{penalty}(q, d), & q \neq d \end{cases}$$

Funcția f calculează scorul între un document și o interogare, fiecare având un singur termen. Ea va fi numită *funcția primitivă de ponderare* (*Primitive weighting function*). Recompensează documentul cu un scor $\text{weight}(q)$ când termenii sunt identici și îi aplică o penalizare în caz contrar.

La pasul inductiv se ia în considerare cazul în care documentul sau interogarea conțin mai mult de un termen.

Pas inductiv: $\forall Q, D$ astfel încât $|Q| \geq 1$ și $|D| \geq 1$,

(1) Presupunem $Q' = Q \cup \{q\}$, atunci $S(Q', D) = S(Q \cup \{q\}, D) = g(S(Q, D), S(\{q\}, D), q, Q, D)$.

(2) Presupunem $D' = D \cup \{d\}$, atunci $S(Q, D') = S(Q, D \cup \{d\}) = h(S(Q, D), S(Q, \{d\}), d, Q, D)$.

Funcția g descrie schimbarea scorului când se adaugă un termen la o interogare și este numită *funcția de creștere a interogării* (*Query growth function*). Când un nou termen q este adăugat la interogarea Q , scorul oricărui document pentru noua interogare ($S(Q \cup \{q\}, D)$) va fi determinat de scorul documentului pentru vechea interogare ($S(Q, D)$), scorul documentului pentru noul termen adăugat ($S(\{q\}, D)$) și orice alte ajustări determinate de D , Q sau q . În mod similar, funcția h descrie schimbarea scorului când se adaugă un termen unui document și este numită *funcția de creștere a documentului* (*Document growth function*).

Este necesară condiția ca $S(Q, D)$ să nu își modifice valorile în funcție de ordinea în care termenii sunt adăugați fie în interogare fie în document.

4.4.2 Constrângeri de regăsire

O altă componentă importantă în modelul axiomatic este reprezentată de constrângerile de regăsire. Au fost definite trei constrângeri pe care orice formulă rezonabilă de calculare a scorului trebuie să le satisfacă.

Constrângerea 1: $\forall Q, D$ și $\forall d \in T$, dacă $d \in Q$, $S(Q, D \cup \{d\}) > S(Q, D)$. Această constrângere spune că adăugarea unui termen din interogare la un document trebuie să crească scorul.

Constrângerea 2: $\forall Q, D$ si $\forall d \in T$, daca $d \notin Q, S(Q, D \cup \{d\}) < S(Q, D)$. Această constrângere asigură că adgarea unui termen care nu face parte dintr interogare la un document are ca urmare scăderea scorului.

Constrângerea 3: $\forall Q, D$ si $\forall d \in T$, daca $d \in Q, \delta_d(d, D, Q) > \delta_d(d, D \cup \{d\}, Q)$, unde $\delta_d(d, D, Q) = S(Q, D \cup \{d\}) - S(Q, D)$. Această constrângere asigură că cantitatea de creștere a scorului prin adăugarea unui termen d din interogare în document trebuie să se micșoreze pe măsură ce se adaugă mai mulți termeni.

4.4.3 Modelarea funcțiilor

Pentru a obține o relație între funcțiile de regăsire deja existente și noul model axiomatic, au fost rescrise câteva din acestea folosind schema inductivă prezentată mai sus. În continuare este prezentată rescrierea formulei *Okapi BM25*.

Okapi definește scorul între o interogare și un document astfel:

$$S(Q, D) = \sum_{t \in Q \cap D} \ln \frac{N - df(t) + 0.5}{df(t) + 0.5} \times QTF(C_t^Q) \times TF_LN(C_t^D, |D|),$$

unde C_t^Q și C_t^D reprezintă numărul de ocurențe ale termenului t în interogarea Q , respectiv în documentul D , $df(t)$ reprezintă numărul de documente care conțin termenul t , $QTF(x) = \frac{(k_3+1) \times x}{k_3+x}$ și $TF_LN(x, y) = \frac{(k_1+1) \times x}{k_1((1-b)+b \frac{y}{avdl})+x}$. k_1 , b și k_3 sunt constante descrise în capitolul anterior.

După rescriere obținem:

$$\begin{aligned} weight(q) &= \ln \frac{N - df(t) + 0.5}{df(t) + 0.5} \cdot TF_LN(1, 1) \\ penalty() &= 0 \\ g() &= S(Q, D) + \Delta QTF(C_t^D) \cdot S(q, D) \\ h() &= S(Q, D) + S(Q, d) \cdot \Delta TF(C_t^D, |D| + 1) \cdot \gamma \\ &+ \sum_{t \in Q \cap D} S(Q, t) \cdot \Delta LN(C_t^D, |D|) \cdot \gamma \\ &= \sum_{t \in Q \cap D - d} S(Q, t) \cdot TF_LN(C_t^D, |D| + 1) \cdot \gamma \\ &+ S(Q, d) \cdot TF_LN(C_t^D + 1, |D| + 1) \cdot \gamma, \end{aligned}$$

unde $\Delta TF(x, y) = TF_LN(x + 1, y) - TF_LN(x, y)$, $\Delta LN(x, y) = TF_LN(x, y + 1) - TF_LN(x, y)$, $\Delta QTF(x) = QTF(x + 1) - QTF(x)$ și $\gamma = \frac{1}{TF_LN(1, 1)}$. Se observă că $weight(q)$ este în strânsă legătură cu idf și că $h()$ implementează normalizarea cu lungimea documentului și normalizarea tf .

Funcțiile derivate

Combinând toate posibilitățile celor trei componente s-au obținut șase noi formule [7]:

$$\begin{aligned} \mathbf{F1-LOG(s):} \quad S(Q, D) &= \sum_{t \in Q \cap D} C_t^D \cdot TF(C_t^D) \cdot LN(|D|) \cdot LW(t) \\ \mathbf{F1-EXP(s,k):} \quad S(Q, D) &= \sum_{t \in Q \cap D} C_t^D \cdot TF(C_t^D) \cdot LN(|D|) \cdot EW(t) \\ \mathbf{F2-LOG(s):} \quad S(Q, D) &= \sum_{t \in Q \cap D} C_t^D \cdot TF_LN(C_t^D, |D|) \cdot LW(t) \\ \mathbf{F2-EXP(s,k):} \quad S(Q, D) &= \sum_{t \in Q \cap D} C_t^D \cdot TF_LN(C_t^D, |D|) \cdot EW(t) \\ \mathbf{F3-LOG(s):} \quad S(Q, D) &= \sum_{t \in Q \cap D} C_t^D \cdot TF(C_t^D) \cdot LW(t) - \gamma(|D|, |Q|) \end{aligned}$$

$$\mathbf{F3-EXP}(s,k): S(Q, D) = \sum_{t \in Q \cap D} C_t^D \cdot TF(C_t^D) \cdot EW(t) - \gamma(|D|, |Q|),$$

unde:

$$\begin{aligned} TF(X) &= 1 + \ln(1 + \ln(x)) \\ LW(t) &= \ln \frac{N+1}{df(t)} \\ EW(t) &= \left(\frac{N+1}{df(t)} \right)^k \\ LN(x) &= \frac{avdl + s}{avdl + x \cdot s} \\ TF_LN(x, y) &= \frac{x}{x + s + \frac{s \cdot y}{avdl}} \\ \gamma(x, y) &= \frac{(x - y) \cdot x \cdot s}{avdl} \end{aligned}$$

și $0 \leq s, k \leq 1$.

În [7] se conchide, în urma experimentelor, că **F2-EXP** este mai stabilă, și, per total, o variantă mai bună decât celelalte cinci funcții.

Capitolul 5

Metode de agregare

Considerăm problema combinării ierarhizărilor rezultate din diferite surse. Principalele aplicații includ motoare de meta-căutare, *combinarea metodelor de ierarhizare*, selectarea documentelor pe baza mai multor criterii, îmbunătățirea preciziei căutării prin asocierea cuvintelor [9]. În lucrarea de față agregarea a fost folosită pentru a combina rezultatele diverselor metode de ierarhizare.

Sarcina ierarhizării unei liste de obiecte pe baza uneia sau mai multor criterii este întâlnită în multe situații. Unul dintre principalele scopuri ale acestui efort este identificarea celor mai bune alternative. Când există un singur criteriu (sau "judecător") pentru ierarhizare, sarcina este relativ ușoară. Problema apare când se fac mai multe ierarhizări după criterii diferite și trebuie găsit un "consens" între acestea. Această problemă poartă numele de *rank aggregation problem*.

Voi prezenta mai întâi un model matematic pentru problema agregării și voi trata apoi două metode de agregare.

Fie U un set finit de obiecte numit *univers*. Putem presupune, fără pierderea generalității, că $U = \{1, 2, \dots, |U|\}$ (unde $|U|$ reprezintă cardinalitatea lui U). O ierarhizare peste U este o listă ordonată: $\tau = (x_1 > x_2 > \dots > x_n)$, unde $x_i \in U$ pentru orice $1 \leq i \leq d$, $x_i \neq x_j$ pentru orice $1 \leq i \neq j \leq d$, și $>$ este o relație de ordonare pe $\{x_1, \dots, x_d\}$ numită *criteriu de ordonare*. Pentru un obiect dat $i \in U$ prezent în τ , $\tau(i)$ reprezintă poziția (sau rangul) lui i în τ .

Dacă ierarhia τ conține toate elementele lui U , atunci se numește *listă(ierarhie) completă*. Există situații când anumite obiecte nu sunt ierarhizate de un anumit criteriu; dacă τ conține doar un subset de elemente din universul U , atunci τ se numește *listă parțială*.

5.1 Borda

Metoda *Borda* (după Jean-Charles, chevalier de Borda, May 4, 1733 – February 19, 1799) este o metodă "pozițională", în sensul că atribuie un scor în funcție de poziția pe care un candidat o ocupă în lista fiecărui votant. Ierarhia finală este dată de sortarea descrescătoare după scorul cumulat. Un prim avantaj al metodelor poziționale este că au o complexitate scăzută: pot fi implementate în timp liniar. În același timp satisfac proprietățile de *anonimitate*, *neutralitate* și *consistență*. Cu toate acestea, nu pot satisface *criteriul Condorcet*. De fapt, este posibil să se demonstreze că nici o metodă care asignează ponderi fiecărei poziții și apoi sortează rezultatele aplicând o funcție ponderilor asociate cu fiecare candidat nu satisface principiul Condorcet.

Fie listele complete τ_1, \dots, τ_k , și S mulțimea tuturor candidaților. Pentru fiecare $c \in S$ și fiecare listă τ_i , metoda Borda calculează scorul $B_i(c) =$ numărul de candidați afați sub c în ierarhia τ_i . Scorul total este definit ca:

$$B(c) = \sum_{i=1}^k B_i(c). \quad (5.1)$$

Candidații sunt apoi sortați în ordinea descrescătoare a acestui scor.

Poziție	Candidat	Formulă	Puncte
1	A	(n - 1)	4
2	B	(n - 2)	3
3	C	(n - 3)	2
4	D	(n - 4)	1
5	E	(n - 5)	0

Tabelul 5.1: Scorurile primite de candidații într-o ierarhizare după metoda Borda

Poziție	Candidat	Formulă	Puncte
1	A	(1 / 1)	1.00
2	B	(1 / 2)	0.50
3	C	(1 / 3)	0.33
4	D	(1 / 4)	0.25
5	E	(1 / 5)	0.20

Tabelul 5.2: Borda cu o funcție diferită de scor

5.2 Agregarea Distanță-Rang

Următoarea abordare se bazează pe folosirea unei măsuri de *distanță* (sau similaritate) între ierarhii. Pașii acestei abordări sunt simpli: mai întâi sunt date ierarhiile care trebuiesc combinate, apoi se definește o distanță între o pereche de ierarhii, urmând să se caute o ierarhizare care are proprietatea că minimizează distanțele de la ea la fiecare dintre ierarhiile date. Privită printr-o perspectivă geometrică, această problemă se reduce la găsirea punctului median al multisetului de ierarhii. De aceea aceste tipuri de agregări se mai numesc și ierarhizări mediene.

Problemele care trebuiesc adresate în continuare sunt: *cum să se definească distanță între ierarhii și ce algoritm trebuie folosit pentru a calcula în mod eficient agregarea?*

5.2.1 Distanța-rang

Fie $\sigma = (x_1 > x_2 > \dots > x_n)$ o ierarhie parțială peste U ; spunem că n este lungimea lui σ . Pentru un element $x \in U \cap \sigma$ definim ordinea lui x în ierarhia σ prin $ord(\sigma, x) = |n - \sigma(x)|$. Prin convenție, dacă $x \in U \setminus \sigma$ atunci $ord(\sigma, x) = 0$.

Fie două ierarhii parțiale σ și τ peste același univers. Distanța-rang dintre ele este definită ca:

$$\Delta(\sigma, \tau) = \sum_{x \in \sigma \cup \tau} |ord(\sigma, x) - ord(\tau, x)|. \quad (5.2)$$

Din moment ce pentru orice $x \in U \setminus (\sigma \cup \tau)$ avem $ord(\sigma, x) = ord(\tau, x) = 0$, următoarea egalitate este adevărată:

$$\begin{aligned} \Delta(\sigma, \tau) &= \sum_{x \in \sigma \cup \tau} |ord(\sigma, x) - ord(\tau, x)| = \\ &= \sum_{x \in \sigma \cup \tau} |ord(\sigma, x) - ord(\tau, x)| + \sum_{x \in U \setminus (\sigma \cup \tau)} |ord(\sigma, x) - ord(\tau, x)| = \\ &= \sum_{x \in U} |ord(\sigma, x) - ord(\tau, x)|. \end{aligned}$$

Există două motive pentru care a fost folosită ordinea în loc de rangul propriu-zis. În primul rând s-a considerat că distanța dintre două ierarhii ar trebui să fie mai mare dacă obiectele situate pe primele locuri diferă. Un alt motiv este constituit de faptul că lungimea ierarhiei este și ea importantă: dacă o ierarhie este mai lungă, criteriul care a produs-o se consideră că a efectuat o analiză mai amănunțită asupra obiectelor, așadar, este mai demnă de încredere decât o ierarhie mai scurtă.

Fie un multiset de ierarhii $T = \{\tau_1, \dots, \tau_k\}$. Se definește $\Delta(\sigma, T) = \sum_{\tau \in T} \Delta(\sigma, \tau)$.

5.2.2 Problema rang-agregării

Fie T un multiset de ierarhii $\{\tau_1, \dots, \tau_k\}$. O *agregare distanță-rang* (RDA - rank-distance aggregation) a acestui multiset este o ierarhie σ , peste același unives ca și ierarhiile din T , care minimizează $\Delta(\sigma, T)$. Setul de agregări RD al lui T se notează $agr(T)$.

Orice ierarhie parțială σ de lungime t care minimizează (printre celelalte ierarhii de lungime t) $\Delta(\sigma, T)$ se numește t -agregare a lui T . Este evident că o t -agregare este o RD-agregare dacă distanța față de T este minimă considerând toate valorile lui t . De asemenea, dacă o t -agregare este în $agr(T)$, atunci toate t -agregările sunt în $agr(T)$, pentru un anume t . Aceste două observații demonstrează că $agr(T)$ conține t -agregările care minimizează distanța până la T , peste toate valorile lui t .

Să presupunem că vrem să calculăm agregările pentru un multiset de ierarhii $T = \{\tau_1, \dots, \tau_p\}$, peste universul de obiecte $U = \{1, 2, \dots, n\}$. Definim matricile pătratice n -dimensionale $D^{(t)}$, $1 \leq t \leq n$ în felul următor:

$$D^{(t)}(k, j) = \begin{cases} \sum_{i=1}^p |j - ord(\tau_i, k)|, & j \leq t \\ \sum_{i=1}^p |ord(\tau_i, k)|, & t < j \end{cases}. \quad (5.3)$$

Fie $\pi = (i_1 > \dots > i_t)$ o ierarhie de lungime t , și, i_{t+1}, \dots, i_n sunt obiectele din U care nu apar în π (adică, $ord(\pi, i_j) = 0, \forall j > t$). Au loc următoarele egalități:

$$\begin{aligned} \Delta(\pi, T) &= \sum_{\tau_i \in T} \Delta(\pi, \tau_i) = \sum_{\tau_i \in T} \sum_{j \in U} |ord(\pi, j) - ord(\tau_i, j)| = \\ &= \sum_{\tau_i \in T} \sum_{j=1, n} |ord(\pi, j) - ord(\tau_i, j)| = \sum_{\tau_i \in T} \sum_{j=1, n} |ord(\pi, i_j) - ord(\tau_i, i_j)| = \\ &= \sum_{\tau_i \in T} \sum_{j=1, t} |j - ord(\tau_i, i_j)| + \sum_{j=t+1, n} |ord(\tau_i, i_j)| = \\ &= \sum_{j=1, n} D^{(t)}(i_j, j). \end{aligned}$$

Așadar, distanța de la π la multisetul T este $\Delta(\pi, T) = \sum_{j=1, n} D^{(t)}(i_j, j)$.

Egalitatea de mai sus este folositoare pentru a găsi o t -agregare: se caută o permutare (i_1, \dots, i_n) a lui U , astfel încât $E = \sum_{j=1, n} D^{(t)}(i_j, j)$ să fie minim; odată găsită o astfel de permutare, t -agregarea este $(i_1 > i_2 > \dots > i_t)$. Pentru a găsi toate t -agregările, se caută toate permutările care minimizează expresia E . Următorul pas este găsirea RD-agregărilor selectând t -agregările care minimizează distanța la multisetul T , pentru orice valoare a lui t .

Găsirea unei t -agregări se reduce la următoarea problemă de optimizare: având o matrice pătratică n -dimensională $M = (m_{i,j})_{1 \leq i, j \leq n}$ cu elemente întregi pozitive, să se găsească mulțimea

$$A = \{(i_1, \dots, i_n) | (i_k \neq i_j \forall k \neq j), (1 \leq i_j \leq n), \text{ și suma } \sum_{j=1, n} m_{i_j, j} \text{ este minimă}\}. \quad (5.4)$$

Soluția clasică a acestei probleme este *Algoritmul Ungar* a cărui prezentare nu ține de scopul acestei lucrări.

Capitolul 6

Studiu comparativ

În cadrul acestei lucrări, ca parte practică, am testat și comparat performanțele câtorva metode de ierarhizare prezentate în capitolele anterioare: *VSM*, *BM25* și *F2EXP*. De asemenea, am implementat și testat o metodă euristică cu scopul de a micșora timpul de căutare; este vorba de *cluster pruning*. Am implementat două tehnici de agregare (cele două prezentate în capitolul anterior) cu scopul de a combina rezultatele obținute cu cele trei funcții de regăsire. Rezultatele testelor sunt prezentate în ultima secțiune a acestui capitol.

Pentru a implementa și testa aceste tehnici, am dezvoltat un "framework" de test folosind diverse unelte prezentate în secțiunea imediat următoare. De asemenea, corpusurile de test sunt prezentate tot aici.

6.1 Unelte

6.1.1 Apache Lucene

Apache Lucene este o bibliotecă software performantă de regăsirea informației scrisă în întregime în *Java* [10] și distribuită gratuit sub licența *open-source Apache*. Pachetul *Lucene* este folosit cu succes de foarte multe companii software în produse atât comerciale cât și open-source, printre ele numărându-se site-ul *Wikipedia* și mediul integrat de dezvoltare *Eclipse IDE*.

În contextul lucrării actuale, *Lucene* a fost folosit pentru a testa și compara diverse metode de ierarhizare, și, ca urmare, voi prezenta în continuare metoda implicită pe care *Lucene* o folosește la ierarhizarea documentelor.

Similaritate și ierarhizare

Lucene combină modelul boolean (BM) cu modelul de spațiu vectorial (VSM): documentele care trec de BM sunt etichetate cu un scor de către VSM.

În VSM, documentele și interogările sunt reprezentate ca vectori de ponderi într-un spațiu multidimensional, unde fiecare termen din index este o dimensiune și ponderile sunt valorile *tf-idf*. VSM nu necesită faptul ca ponderile să fie valori *tf-idf*, dar aceste ponderi au rezultate foarte bune în practică, și, ca urmare, *Lucene* folosește această abordare. Pentru un termen t și un document (sau interogare) x , $tf(t,x)$ crește odată cu numărul de ocurențe ale lui t în x iar $idf(t)$ descrește odată cu creșterea numărului de documente din index care îl conțin pe t .

Scorul documentului d pentru interogarea q este dat de *similaritatea cosinus* pentru vectorii de ponderi $V(q)$ și $V(d)$:

$$cos-sim(q, d) = \frac{V(q)V(d)}{|V(q)||V(d)|},$$

unde numărătorul reprezintă produsul scalar, iar numitorul, produsul normelor euclidiene. Ecuația poate fi văzută și ca produsul scalar dintre cei doi vectori normalizați.

Lucene perfecționează VSM atât în materie de calitate cât și de uzabilitate.

- Normalizarea lui $V(d)$ la vectorul unitate poate pune unele probleme în sensul că îndepărtează toată informația despre lungimea documentului. Pentru unele documente, lucrul acesta poate reprezenta o problemă. Pentru a evita această problemă, Lucene folosește un alt factor de normalizare a lungimii documentului, care normalizează vectorul la un vector mai mare sau egal decât vectorul unitate: $\text{doc-len-norm}(d)$.
- La indexare utilizatorii pot specifica faptul că unele documente sunt mai importante decât altele prin asignarea unui *boost* respectivelor documente. Ca urmare, scorul fiecărui documente este multiplicat cu această valoare: $\text{doc-boost}(d)$.
- Lucene este bazat pe câmpuri (secțiuni ale unui document), și, ca urmare, fiecare termen al unei interogări se aplică unui singur câmp, normalizarea vectorului se aplică la nivel de câmp, și se pot specifica și nivele de boost pentru câmpuri.
- Același câmp poate fi adăugat unui document în timpul indexării de mai multe ori, iar, ca urmare, nivelul de boost al acelui câmp este dat de înmulțirea nivelelor de boost ale adăugărilor.
- La căutare utilizatorii pot specifica nivele de boost pentru fiecare interogare, sub-interogare și termen al unei interogări.
- Un document poate fi relevant la o interogare cu mai mulți termeni fără să contină toți termenii prezenți în interogare, iar documentele în care apar mai mulți termeni pot fi "răsplătite" printr-un factor de coordonare, care este mai mare când mai mulți termeni sunt prezenți: $\text{coord-factor}(q, d)$.

Făcând asumția simplificatoare că există un singur câmp în index, *formula conceptuală de scor* pentru Lucene este următoarea:

$$\text{score}(q, d) = \text{coordfactor}(q, d) \times \text{queryboost}(q) \times \frac{V(q) \times V(d)}{|V(q)|} \times \text{doclennorm}(d) \times \text{docboost}(d)$$

Din această formulă se derivează *formula practică de scor* care este implementată de Lucene. Pentru calcularea eficientă a scorului, unele componente sunt calculate și agregate la indexare:

- Nivelul de boost pentru interogare este cunoscut când căutarea începe.
- Norma euclidiană a vectorului interogare poate fi calculată când începe căutarea, dat fiind faptul că e independentă de documentul pentru care se calculează scorul la un moment dat. Din perspectiva optimizării, merită pusă întrebarea: *are rost să se normalizeze vectorul interogării, din moment ce toate scorurile vor fi multiplicare cu aceeași valoare?* Ca urmare, ierarhia documentelor pentru o interogare dată nu va fi afectată de normalizare. Există două motive pentru a păstra normalizarea:
 - scorurile unui document pentru interogări distincte trebuie să fie comparabile (într-o anumită măsură)
 - aplicarea normalizării păstrează scorurile "în jurul" vectorului unitate, împiedicând astfel alterarea scorurilor din cauza limitării de precizie ale numerelor în virgulă mobilă
- Norma pentru fiecare document $\text{doc-len-norm}(d)$ și nivelul de boost $\text{doc-boost}(d)$ sunt cunoscute la indexare. Sunt calculate și rezultatul înmulțirii lor este salvat ca o singură valoare în index: $\text{norm}(d)$.

În continuare este prezentată formula practică de scor:

$$\text{score}(q, d) = \text{coord}(q, d) \times \text{queryNorm}(q) \times \sum_{t \in q} (tf(t, d) \times idf(t)^2 \times \text{boost}(t) \times \text{norm}(\text{field}(t), d)),$$

unde:

1. $tf(t, d)$ este corelat cu frecvența termenului în document. Documentele în care un termen apare de mai multe ori primesc un scor mai mare pentru acel termen. Lucene implementează astfel: $tf(t, d) = \sqrt{\text{freq}}$, unde freq reprezintă de câte ori apare termenul în document.

2. $idf(t)$ este inversul frecvenței termenului la nivel de index. Acest lucru înseamnă că termenii mai rari au o contribuție mai mare la scor. Implementarea Lucene este: $idf(t) = 1 + \log(\frac{numDocs}{docFreq+1})$, unde $numDocs$ reprezintă numărul de documente din index și $docFreq$ reprezintă numărul de documente în care apare termenul.
3. $coord(q, d)$ este o componentă calculată la momentul căutării: $coord(q, d) = \frac{overlap}{maxOverlap}$, unde $overlap$ reprezintă numărul de termeni din interogare care se regăsesc în document și $maxOverlap$, numărul de termeni ai interogării.
4. $queryNorm(q)$ este factorul de normalizare folosit pentru a face scorurile pentru diferite interogări comparabile. Acest factor nu afectează ierarhizarea documentelor din moment ce este același pentru fiecare document. Implementarea implicită Lucene computează norma euclidiană a vectorului ponderilor (ajustate de nivelele de boost):

$$queryNorm(q) = \frac{1}{\sqrt{boost(q) \times \sum_{t \in q} (idf(t) \times boost(t))^2}}$$

5. $boost(t)$ reprezintă nivelul de boost al termenului; acesta poate fi setat din sintaxa interogării dacă este folosit parserul de interogări pus la dispoziție de Lucene, sau prin intermediul api-ului obiectului *Query*.

Pachetul contrib/benchmark/quality

Proiectul *Lucene Java* conține și un "spațiu de lucru" numit *Lucene Contrib* care găzduiește contribuții "third-party", contribuții cu dependențe externe (pachetul principal nu are dependențe) și implementări de noi idei.

Printre contribuțiile din acest set de pachete se află pachetul *analyzers* ce conține o multitudine de analizoare (componente de preprocesare a textului: liste de stop, stemmer-e) și pachetul *benchmark*. Pachetul *benchmark* conține unelte pentru testarea și evaluarea lui *Lucene* folosind corpusuri standard. În speța, subpachetul *quality* este folosit la rularea unui set de interogări în format standard (de exemplu TREC) pe un index de documente și la calcularea unor măsuri de evaluare a sistemului.

6.1.2 Proiectul Apache Open Relevance (ORP)

Proiectul *ORP (Open Relevance Project)* este un sub-proiect Lucene care are scopul de a construi materiale pentru evaluarea relevanței în regăsirea informației (de asemenea, *Machine Learning* și *Procesarea limbajului natural*) urmând ca acestea să fie distribuite sub licență open-source[12]. Aceste materiale sunt compuse din colecții de documente, unul sau mai multe seturi de interogări și judecăți de relevanță pentru fiecare set.

În timp ce *TREC* și alte conferințe pun la dispoziție corpusuri, seturi de interogări și judecăți de relevanță, niciuna dintre ele nu face acest lucru în mod "gratuit" și "deschis". Intrarea în posesia unei astfel de colecții implică de obicei, o sumă de bani, semnarea unui contract și livrarea prin poștă.

În momentul de față *ORP* pune la dispoziție trei corpusuri de test: *Ohsumed*, *Hamshahri* și *Tempo*. *ORP* conține o serie de scripturi *ant* (<http://ant.apache.org/>) și adaptoare (programe Java) folosite pentru a descărca de pe Internet respectiv converti formatele originale ale acestor colecții la un format standard (TREC). În continuare voi prezenta fiecare dintre aceste colecții.

Ohsumed

Acest corpus este versiunea folosită la *TREC-9 filtering track*. Corpusul poate fi obținut de la adresa http://trec.nist.gov/data/t9_filtering.html.

Colecția de test *Ohsumed* este un set de 348,556 referințe din *MEDLINE* (baza de date a *Bibliotecii Naționale de Medicină a Statelor Unite*), constând din titluri și/sau rezumate din 270 de jurnale medicale (1987-1991). Câmpurile disponibile sunt *title*, *abstract*, *MeSH indexing terms*, *author*, *source*, și *publication type*.

Colecția pune la dispoziție trei seturi diferite de topicuri (interogări):

1. un subset de 63 de interogări din setul original.
2. un set de 4904 de termeni *MeSH** și definițiile lor(MSH)
3. un subset de 500 de termeni *MeSH* (MSH-SMP)

(*)*MeSH* este vocabularul "controlat" al Bibliotecii Naționale de Medicină a Statelor Unite folosit la indexarea articolelor pentru MEDLINE. Terminologia MeSH asigură un mod consistent de a regăsi informație care poate folosi terminologii diferite pentru aceleași concepte.

Hamshahri

Corpusul *Hamshahri* conține documente de știri din ziarul on-line în limba persană *Hamshahri* (<http://www.hamshahrionline.ir/>). Colecția se găsește la adresa <http://ece.ut.ac.ir/dbrg/Hamshahri/> și conține documente din 1996 până în 2002, acoperind 82 de categorii diferite (politică, literatură, artă, etc.).

Colecția conține două seturi de interogări de câte 65 respectiv 58 de subiecte și sunt cunoscute următoarele statistici:

- Mărime(MB): 345MB(564MB cu etichete)
- Număr de documente: 166,774
- Număr de termeni: 417,339
- Lungimea medie a unui document (în cuvinte): 380.

Mai multe informații se pot găsi la în articolul *Hamshahri: A Standard Persian Text Collection* (http://ece.ut.ac.ir/dbrg/Hamshahri/Papers/Hamshahri_Description.pdf).

Tempo

Corpusul *Tempo* conține articole de știri din perioada 2000-2002 din ziarul indonezian on-line *Tempo* (<http://www.tempointeraktif.com/>). Colecția se poate descărca de la adresa <http://ilps.science.uva.nl/resources/bahasa>.

Corpusul are următoarele statistici:

- Mărime(MB): 45.57MB
- Număr de documente: 22,944
- Media de termeni unici: 155
- Lungimea medie a unui document (în octeți): 1549.59.

Există un set de 35 de interogări care au ca subiect evenimente petrecute în Indonezia în acea perioadă. Statisticile interogărilor sunt următoarele:

- Număr de interogări: 35
- Lungimea medie(în cuvinte) 5.2
- Media de cuvinte unice: 5.17
- Media numărului de documente relevante per interogare: 66.971.

6.1.3 trec_eval

trec_eval este un standard folosit de comunitatea TREC pentru a evalua sisteme ad-hoc de regăsirea informației, folosind un fișier standard de judecăți de relevanță și fișierul cu rezultatele sistemului și calculând diferite măsuri de evaluare.

Majoritatea opțiunilor pot fi ignorate, singura folosită mai des este "-q", care, în cazul în care este specificată, duce la afișarea rezultatelor pentru toate interogările, nu doar mediile. Un exemplu de utilizare oficială poate fi:

```
trec_eval -q -c -M1000 official_qrels submitted_results.
```

pentru a asigura evaluarea corectă dacă fișierul de rezultate nu conține rezultate pentru toate interogările, sau conține mai mult de 1000 de rezultate per interogare.

Folosire:

```
trec_eval [-h] [-q] [-a] [-o] [-c] [-l<num>] [-N<num>] [-M<num>] [-Ua<num>] [-Ub<num>]
[-Uc<num>] [-Ud<num>] [-T] trec_rel_file trec_top_file
```

Există mai multe opțiuni care pot fi specificate la o rulare *trec_eval*. În continuare sunt prezentate o parte din ele.

-h: printează mesajul de help și termină

-q: pe lângă sumarul evaluărilor, printează rezultatele pentru evaluarea fiecărei interogări

-a: printează toate măsurile, în loc de măsurile oficiale pentru TREC

-o: printează în formatul nonrelațional(implicit este relațional)

-c: media este făcută peste setul complet de interogări din judecățile de relevanță în loc de intersecția dintre judecăți și rezultatele sistemului. Interogările care lipses vor contribui valoarea 0 la toate măsurile de evaluare(acceptabil pentru măsurile TREC standard, nu neapărat și pentru celelalte)

trec_eval citește tupluri în următorul format din fișierul rezultatelor:

```
qid iter docno rank sim run_id
```

Acest tuplu reprezintă un document cu numărul *docno* regăsit în contextul interogării cu id-ul *qid* cu scorul similarității *sim*. Celelalte câmpuri sunt ignorate, cu excepția câmpului *run_id* care este printat. Câmpul *rank* este ignorat, rangurile fiind calculate prin sortarea după câmpul *sim*(problema egalităților de scoruri între documente este rezolvată deterministic, folosind *docno*). Așadar *sim* se presupune că este mai mare pentru documentele mai relevante.

Relevanța unui document *docno* la o interogare *qid* este dată de tuplurile din fișierul cu judecăți de relevanță.

```
qid iter docno rel,
```

Un tuplu reprezintă relevanța *rel* (întreg pozitiv mai mic decât 128, sau -1(nejudecat)) a unui document *docnum* la o interogare *qid*. Câmpul *iter* este ignorat. Interogările pentru care nu există informație despre relevanță sunt ignorate. Interogările pentru care există documente relevante dar nu și documente regăsite sunt ignorate implicit; acest lucru permite sistemelor să evalueze pe subseturi ale documentelor relevante, dar dacă un sistem nu întoarce nici un documente pentru o interogare, acest lucru nu va afecta sumarul măsurilor. Pentru a modifica acest comportament se folosește opțiunea -c.

Formatul de output al *trec_eval* este:

```
measure_name query value,
```

unde *measure_name* reprezintă numele măsurii, *query* reprezintă id-ul interogării (în cazul mediilor, valoare *all* este afișată) și *value* este valoarea măsurii. Tabelul 6.1 prezintă numele măsurilor oficiale TREC și un scurt sumar.

num_ret	Numărul total de documente regăsite
num_rel	Numărul total de documente relevante
num_rel_ret	Numărul total de documente relevante și regăsite
map	Precizia medie (MAP)
gm_ap	Precizia medie calculată folosind media geometrică
R-prec	R-precizia (precizia după R documente regăsite)
bpref	Preferința binară
recip_rank	Rangul reciproc al primului document relevant
ircl_prn.0.00	Media preciziei interpolate la nivelul de recall 0.00
ircl_prn.0.10	Media preciziei interpolate la nivelul de recall 0.10
ircl_prn.0.20	Media preciziei interpolate la nivelul de recall 0.20
ircl_prn.0.30	Media preciziei interpolate la nivelul de recall 0.30
ircl_prn.0.40	Media preciziei interpolate la nivelul de recall 0.40
ircl_prn.0.50	Media preciziei interpolate la nivelul de recall 0.50
ircl_prn.0.60	Media preciziei interpolate la nivelul de recall 0.60
ircl_prn.0.70	Media preciziei interpolate la nivelul de recall 0.70
ircl_prn.0.80	Media preciziei interpolate la nivelul de recall 0.80
ircl_prn.0.90	Media preciziei interpolate la nivelul de recall 0.90
ircl_prn.1.00	Media preciziei interpolate la nivelul de recall 1.00
P5	Precizia după 5 documente regăsite
P10	Precizia după 10 documente regăsite
P15	Precizia după 15 documente regăsite
P20	Precizia după 20 documente regăsite
P30	Precizia după 30 documente regăsite
P100	Precizia după 100 documente regăsite
P200	Precizia după 200 documente regăsite
P500	Precizia după 500 documente regăsite
P1000	Precizia după 1000 documente regăsite

Tabelul 6.1: Măsurile oficiale afișate de trec_eval

6.2 Framework-ul de test

Pachetul `/contrib/benchmark/quality` nu a fost de ajuns pentru a automatiza procesul de testare și a extrage statisticile pe care le-am considerat necesare. Ca urmare, am dezvoltat un "framework", folosind câteva unelte (majoritatea sunt descrise mai sus), care are următoarele facilități:

1. se pot adăuga colecții de test compuse din: un set de documente, seturi de interogări și relevanțe (standardul TREC);
2. se pot implementa metode de ierarhizare ca extensii la *Lucene* după o anumită convenție;
3. se poate rula un test al unei metode pe un anumit corpus, cu diverse opțiuni (setul de interogări, câmpurile interogării, *liste de stop/stemmer*, etc.);
4. se poate folosi o metodă de agregare pentru a combina rezultatele mai multor tehnici de ierarhizare;
5. după testarea mai multor metode de ierarhizare pe un anumit corpus, se poate compila un raport al rezultatelor, compus dintr-un tabel cu diferite măsurători calculate de *trec_eval* și un grafic al preciziei interpolate în 11 puncte de recall; raportul are scopul de a facilita compararea rezultatelor diverselor metode implementate.

Framework-ul este construit folosind următoarele componente:

- *Apache ant* - pentru automatizarea task-urilor;
- *Apache Lucene*, în speță pachetul `/contrib/benchmark` - pentru indexarea documentelor și rularea interogărilor și compilarea fișierului cu rezultate în format standard pentru *trec_eval* (această componentă a fost extinsă pentru a facilita adăugarea de noi metode într-un mod simplu și agregarea rezultatelor mai multor metode folosind o tehnică specificată de agregare);

- *trec_eval* - pentru a calcula măsurile standard de performanță pe baza judecăților de relevanță și a rezultatelor;
- *JFreeChart* (<http://www.jfree.org/jfreechart/>) - pentru a construi grafice *precizie-recall*;
- *Luke* (<http://www.getopt.org/luke/>) - pentru a vizualiza conținutul unui index *Lucene*.

Figura 6.1 ilustrează structura acestui framework. Directorul *benchmark/code/* conține codul *Java* atât al framework-ului propriu-zis cât și al implementărilor de diferite metode de ierarhizare și agregare. Directorul *benchmark/results/* conține rezultatele testelor (fișierele de ieșire *trec_eval* cu nume reprezentative). Tot aici se compilează și rapoartele pe testele efectuate pe un anumit corpus. Directorul *jars/* conține dependențele framework-ului (dacă se implementează o metodă care are pachete externe ca dependențe, acestea vor fi depozitate în acest director). Directorul *utils/* conține anumite utilitare folosite: *trec_eval* și *luke*. În final, fișierul *build.xml* conține scriptul *ant* care automatizează diverse procese prezentate mai jos.

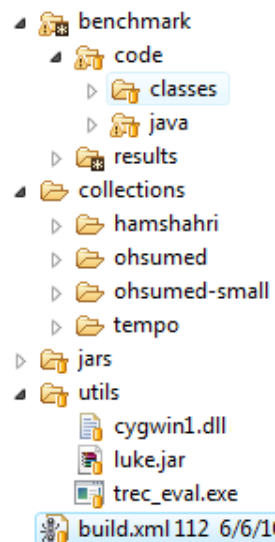


Figura 6.1: Structura framework-ului

Pentru a adăuga o nouă metodă de ierarhizare tot ceea ce trebuie făcut este să se adauge un nou pachet *Java* cu numele *ro.ranking.technique.[NUME-METODĂ]* care să conțină o clasă numită *RankingTechniqueImpl* (trebuie să implementeze interfața *RankingTechnique*). Această clasă furnizează anumite componente folosite de modulul de testare. Modulul de testare nu face decât să folosească componentele *Lucene* obținute prin intermediul clasei pentru a interoga indexul unei colecții și a scrie rezultatele într-un fișier. În momentul în care se efectuează un test, metoda este încărcată dinamic în funcție de parametrii pasați programului. Implementarea unei metode de agregare se face într-un mod similar: trebuie creată o implementare a interfeței *Aggregator* după șablonul *ro.ranking.aggregator.[NUME].AggregatorImpl*.

Procesele de *indexare*, *testare*, *raportare*, *vizualizare index* sunt automatizate folosind următoarele *task-uri ant*:

- **compile:** compilarea codului *java*;
- **index:** indexarea unei colecții cu folosind un anumit analizor (listă de stop, stemmer);
- **benchmark:** testarea unei metode pe o anumită colecție, cu un anumit analizor, folosind un anumit set de interogări (dacă sunt specificate mai multe tehnici, rezultatele lor sunt agregate folosind o anumită metodă de agregare) - toți acești parametri se pot specifica la rulare dar au și valori implicite;
- **report:** compilarea unui raport al performanțelor pe o anumită colecție
- **luke:** vizualizarea conținutului unui anumit index.

6.3 Rezultate

6.3.1 VSM, BM25 și F2EXP

Cele trei funcții de regăsire au avut rezultate destul de apropiate, dar *F2EXP* tinde să aibă rezultate mai bune decât celelalte două funcții candidat. În figurile 6.2, 6.3 și 6.4 se pot compara rezultatele pe cele trei corpusuri. Sigurul corpus la care *BM25* a avut rezultate puțin mai bune decât celelalte metode a fost corpusul indonezian *Tempo*.

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-simple	2	4967	2013945	1164552	816278	0.7035	0.5924	0.6847	0.9455	0.9076	0.8578	0.7036	0.1643
f2exp-T-simple	1	4967	2010658	1164552	853600	0.7535	0.6392	0.7300	0.9457	0.9069	0.8648	0.7281	0.1719
lucene-T-simple	1	4967	2010658	1164552	846109	0.7380	0.6236	0.7131	0.9415	0.9034	0.8586	0.7176	0.1703

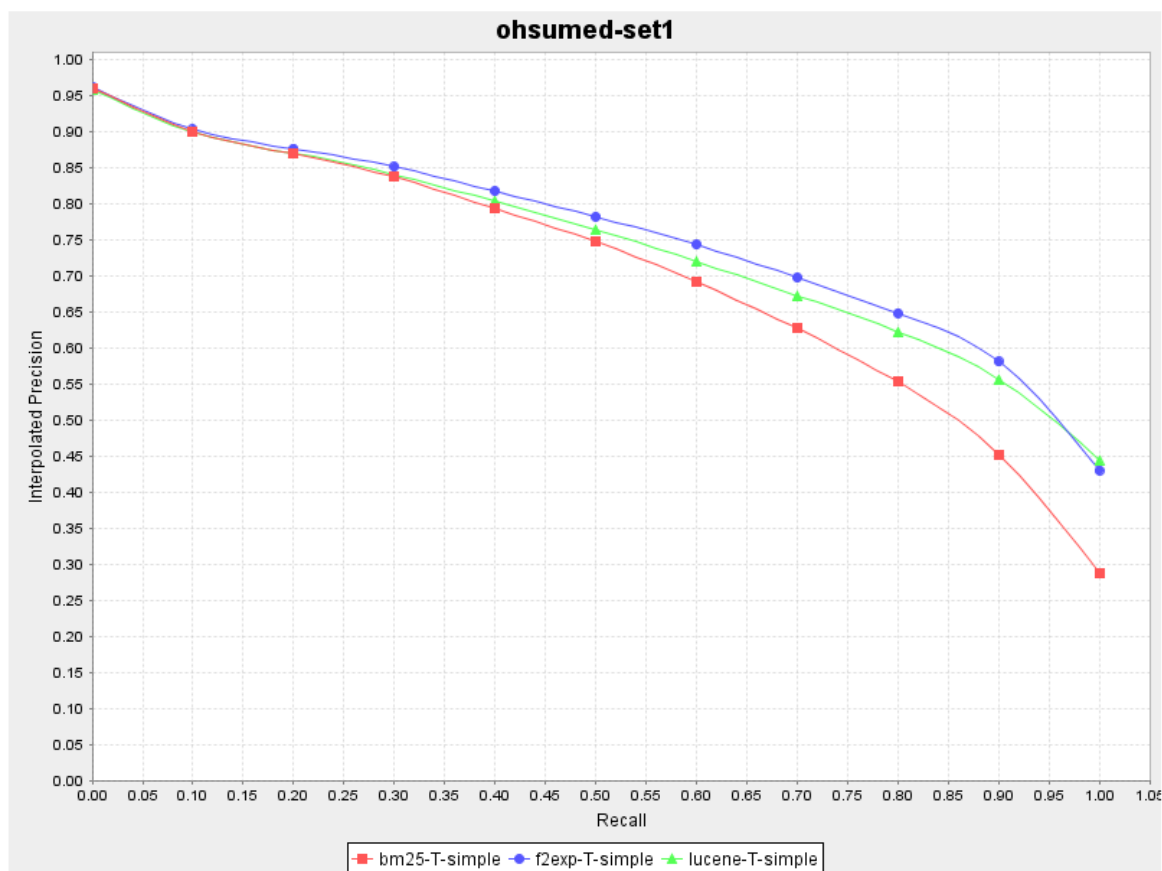


Figura 6.2: Rezultate Lucene VSM, F2EXP și BM25 la rularea pe corpusul *Oshumed* primul set de interogări, folosind doar câmpul *Title* (interogări "scurte")

În figura 6.5 sunt prezentate rezultatele pe setul 2 de interogări a corpusului persan *Hamshahri* pentru interogări formate doar din titlu și interogări formate din titlu și secțiunea *narrative*. Interogările "scurte" au rezultate mai bune.

Figura 6.6 compară rezultatele interogărilor *MeSH* (vezi secțiunea 6.1.2) cu rezultatele interogărilor formate din descrieri amănunțite. Datorită standardului MeSH, primele au rezultate mult mai bune.

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-simple	6	64	32000	2341	2130	0.4016	0.2070	0.3823	0.7468	0.5594	0.4000	0.2466	0.0333
f2exp-T-simple	4	64	32000	2341	2143	0.4445	0.2347	0.4260	0.7763	0.5938	0.4370	0.2622	0.0335
lucene-T-simple	4	64	32000	2341	2126	0.3785	0.1846	0.3650	0.6900	0.4937	0.3865	0.2430	0.0332

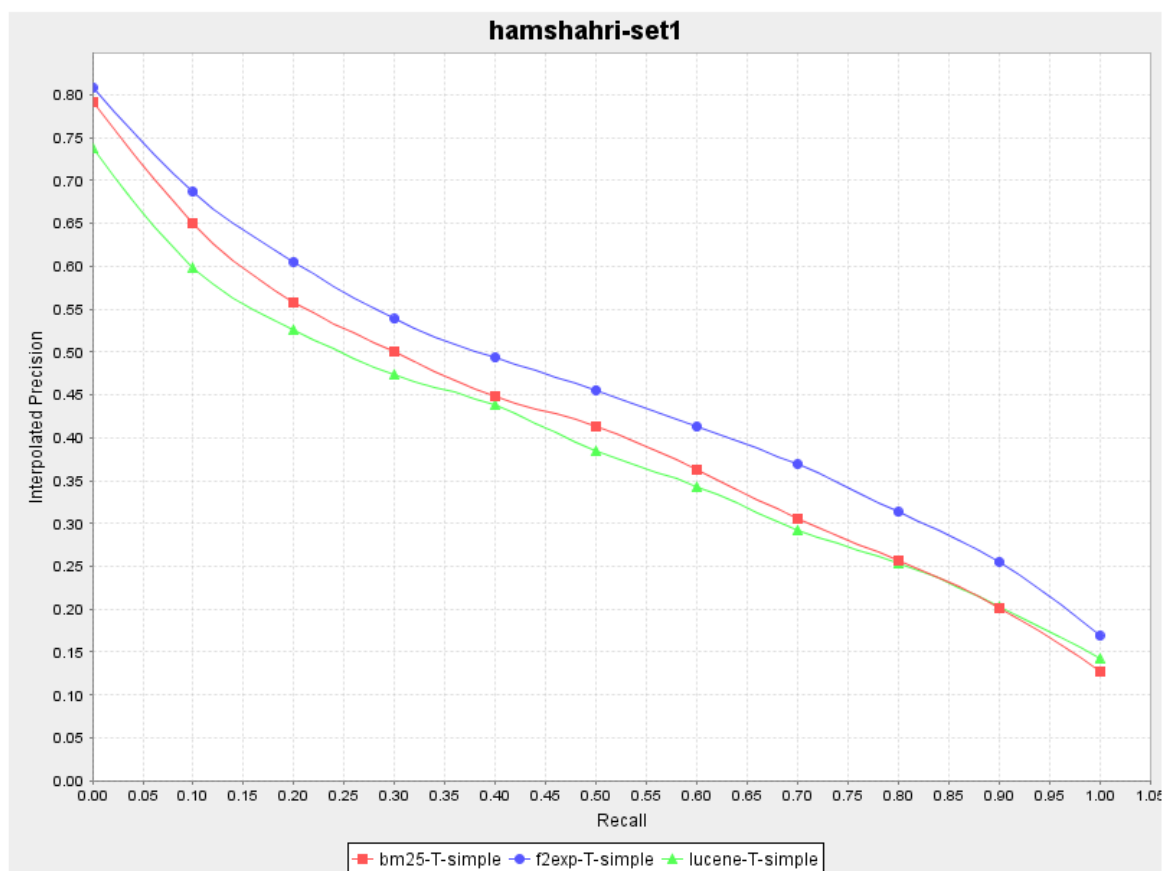


Figura 6.3: Rezultate Lucene VSM, F2EXP și BM25 la rularea pe corpusul *Hamshahri* primul set de întrebări, folosind doar câmpul *Title*

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-simple	8	35	17411	2344	2036	0.5109	0.4623	0.5091	0.7938	0.6171	0.4971	0.3354	0.0582
f2exp-T-simple	5	35	17411	2344	2020	0.5023	0.4498	0.4963	0.8414	0.6000	0.4933	0.3266	0.0577
lucene-T-simple	4	35	17411	2344	2031	0.5033	0.4542	0.5037	0.8167	0.6000	0.4886	0.3311	0.0580

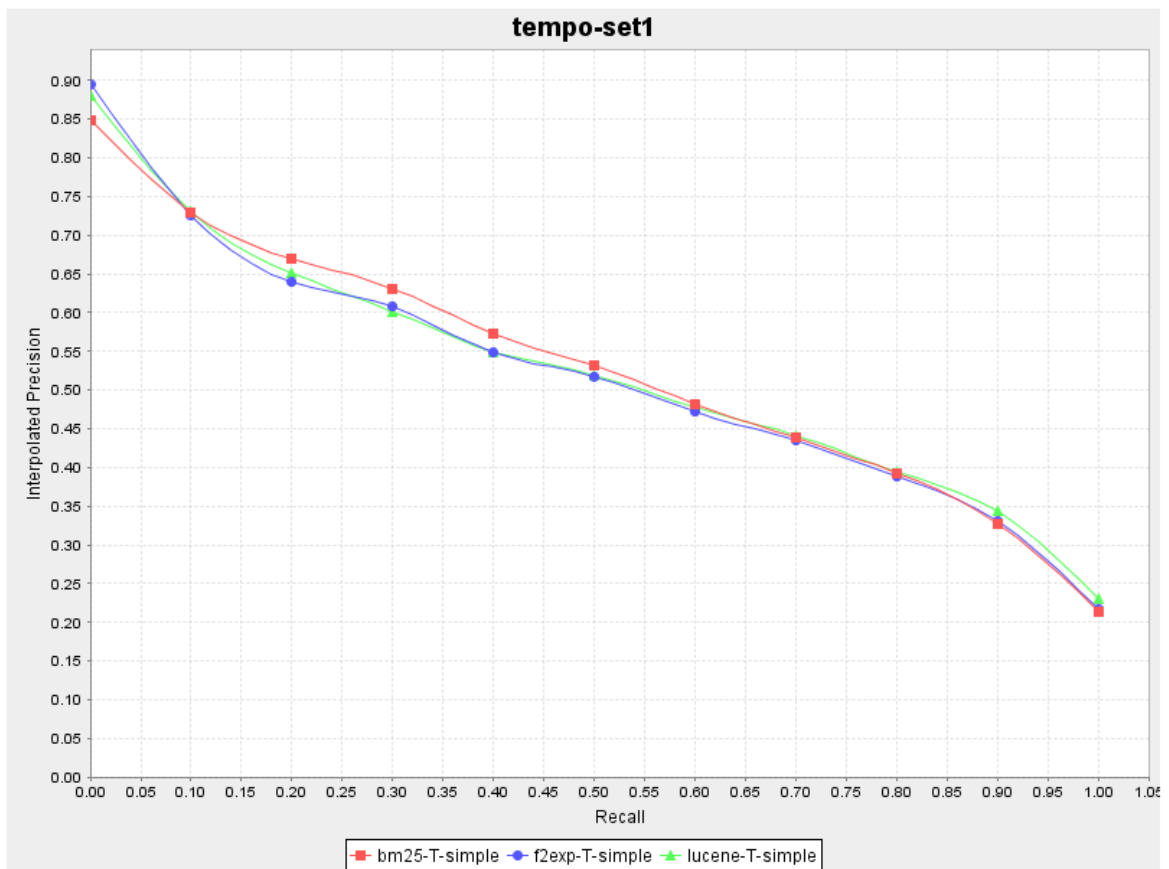


Figura 6.4: Rezultate Lucene, F2EXP și BM25 la rularea pe corpusul *Oshumed*

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-simple	7	54	26523	1629	1103	0.2843	0.1054	0.3006	0.5864	0.4889	0.3049	0.1456	0.0204
bm25-TN-simple	244	57	28500	1725	958	0.1719	0.0530	0.2078	0.5486	0.3579	0.2018	0.1035	0.0168
f2exp-T-simple	5	54	26523	1629	1129	0.2884	0.1151	0.3037	0.6470	0.4852	0.3105	0.1428	0.0209
f2exp-TN-simple	189	57	28500	1725	879	0.1560	0.0425	0.1868	0.5373	0.3158	0.1766	0.0925	0.0154
lucene-T-simple	5	54	26523	1629	1107	0.2902	0.0966	0.3058	0.6023	0.4889	0.3099	0.1476	0.0205
lucene-TN-simple	280	57	28500	1725	1096	0.2436	0.0719	0.2622	0.6053	0.4316	0.2509	0.1326	0.0192

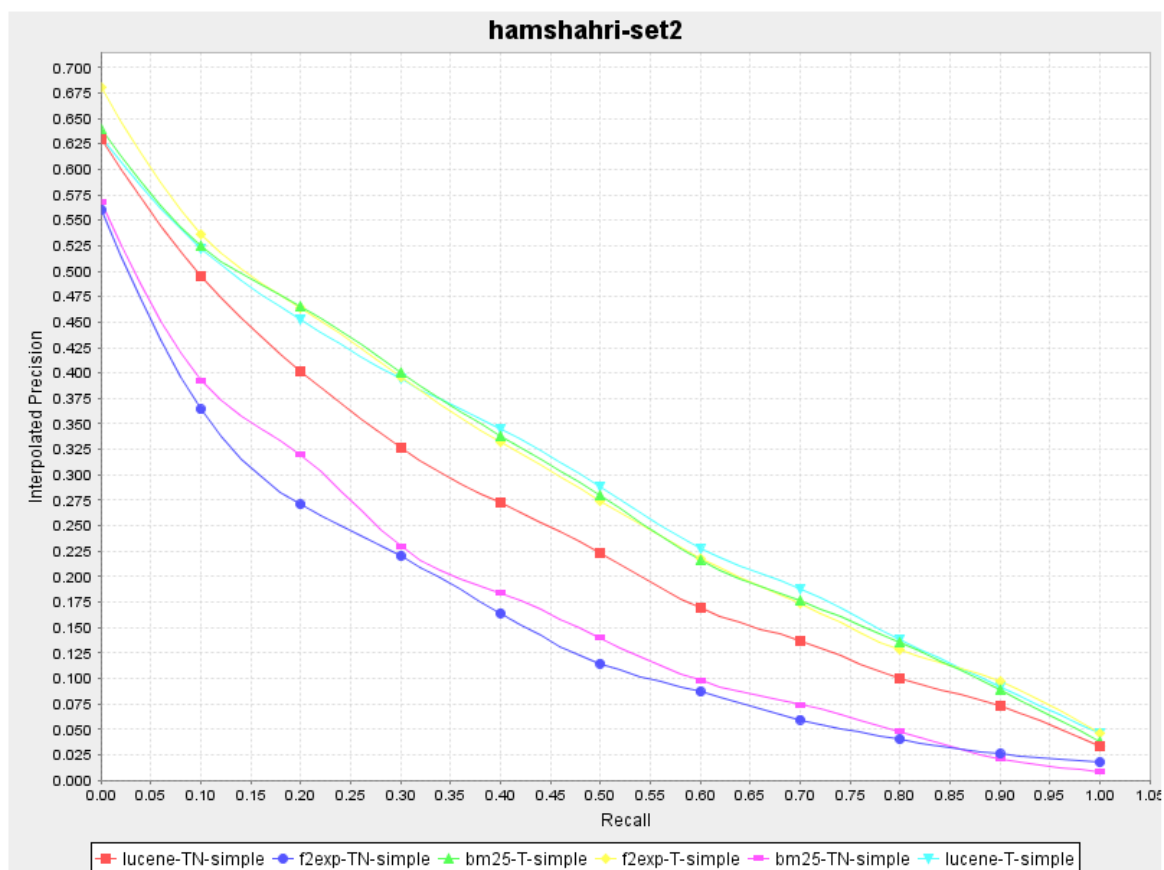


Figura 6.5: Interogări "lungi" și interogări "scurte" (titlu + descriere și titlu), setul doi de interogări, corpus *Hamshahri*

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-D-simple	256	4904	2449158	1161347	190474	0.0678	0.0096	0.1183	0.4231	0.2636	0.1979	0.1453	0.0388
bm25-T-simple	2	4904	1982445	1161347	815077	0.7116	0.6284	0.6923	0.9544	0.9176	0.8675	0.7117	0.1662
f2exp-D-simple	102	4904	2448859	1161347	187518	0.0706	0.0111	0.1233	0.4352	0.2721	0.2082	0.1498	0.0382
f2exp-T-simple	1	4904	1979633	1161347	852341	0.7621	0.6776	0.7380	0.9550	0.9169	0.8745	0.7363	0.1738
lucene-D-english	15	4904	2449499	1161347	243731	0.1004	0.0179	0.1551	0.4618	0.2989	0.2381	0.1828	0.0497
lucene-D-simple	73	4904	2448859	1161347	216281	0.0845	0.0136	0.1373	0.4431	0.2840	0.2207	0.1642	0.0441
lucene-T-simple	1	4904	1979633	1161347	844848	0.7464	0.6615	0.7209	0.9505	0.9135	0.8683	0.7258	0.1723

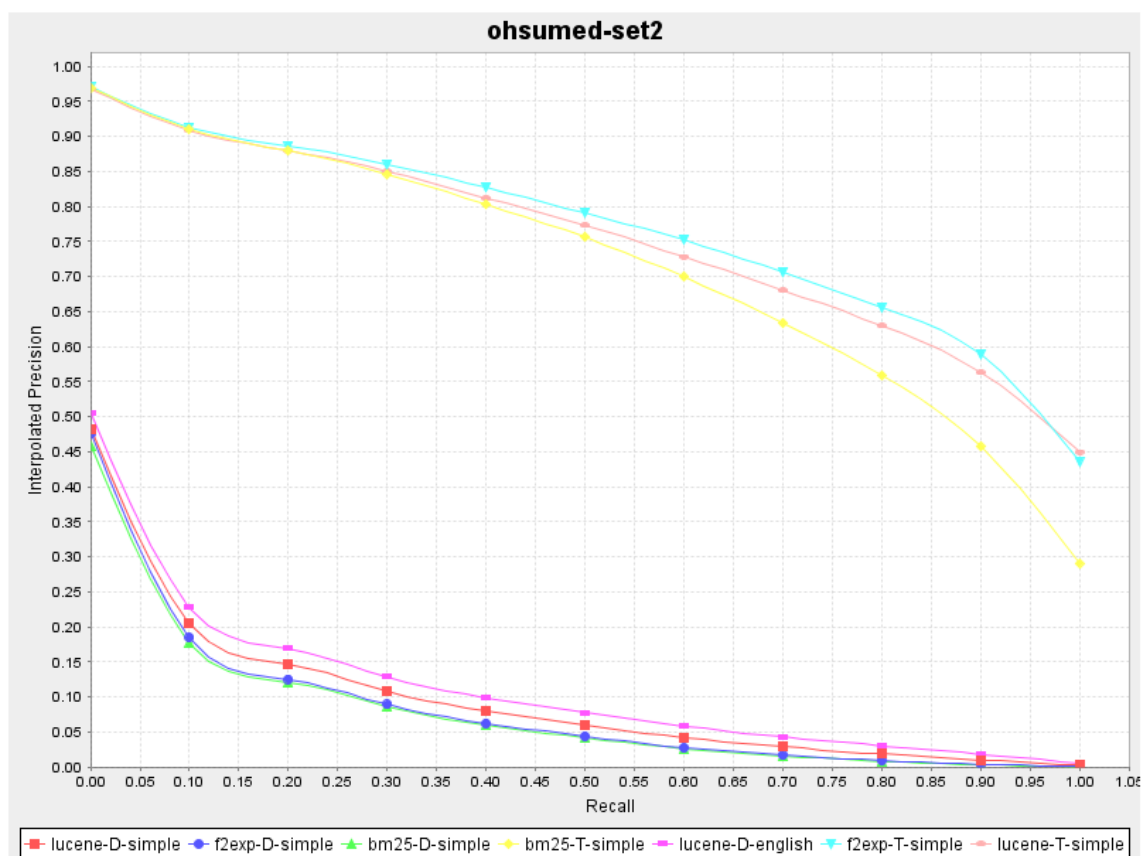


Figura 6.6: Rezultate pentru interogările *MeSH*(titlu vs. descriere)

Concluzia ultimelor două experimente este că interogările care conțin doar cuvintele esențiale au rezultate mult mai bune.

6.3.2 Preprocesarea textului

În continuare voi prezenta rezultatele testelor efectuate folosind liste de stop și stemmer-e pentru colecțiile *Hamshahri* (6.7) și *Ohsumed* (6.8). La primul corpus rezultatele celor trei funcții cresc similar (nu diferă în funcție de metoda de ierarhizare) față de rezultatele inițiale. Dar, în mod ciudat, pe corpusul englez, rezultatele sunt mai slabe decât cele inițiale.

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-persian	6	65	32208	2352	2314	0.4851	0.4157	0.4548	0.8616	0.6646	0.4636	0.2740	0.0356
bm25-T-simple	6	64	32000	2341	2130	0.4016	0.2070	0.3823	0.7468	0.5594	0.4000	0.2466	0.0333
f2exp-T-persian	4	65	32208	2352	2324	0.5326	0.4739	0.5006	0.8933	0.7015	0.5041	0.2949	0.0358
f2exp-T-simple	4	64	32000	2341	2143	0.4445	0.2347	0.4260	0.7763	0.5938	0.4370	0.2622	0.0335
lucene-T-persian	5	65	32208	2352	2326	0.4805	0.4182	0.4514	0.8369	0.6308	0.4692	0.2783	0.0358
lucene-T-simple	4	64	32000	2341	2126	0.3785	0.1846	0.3650	0.6900	0.4937	0.3865	0.2430	0.0332

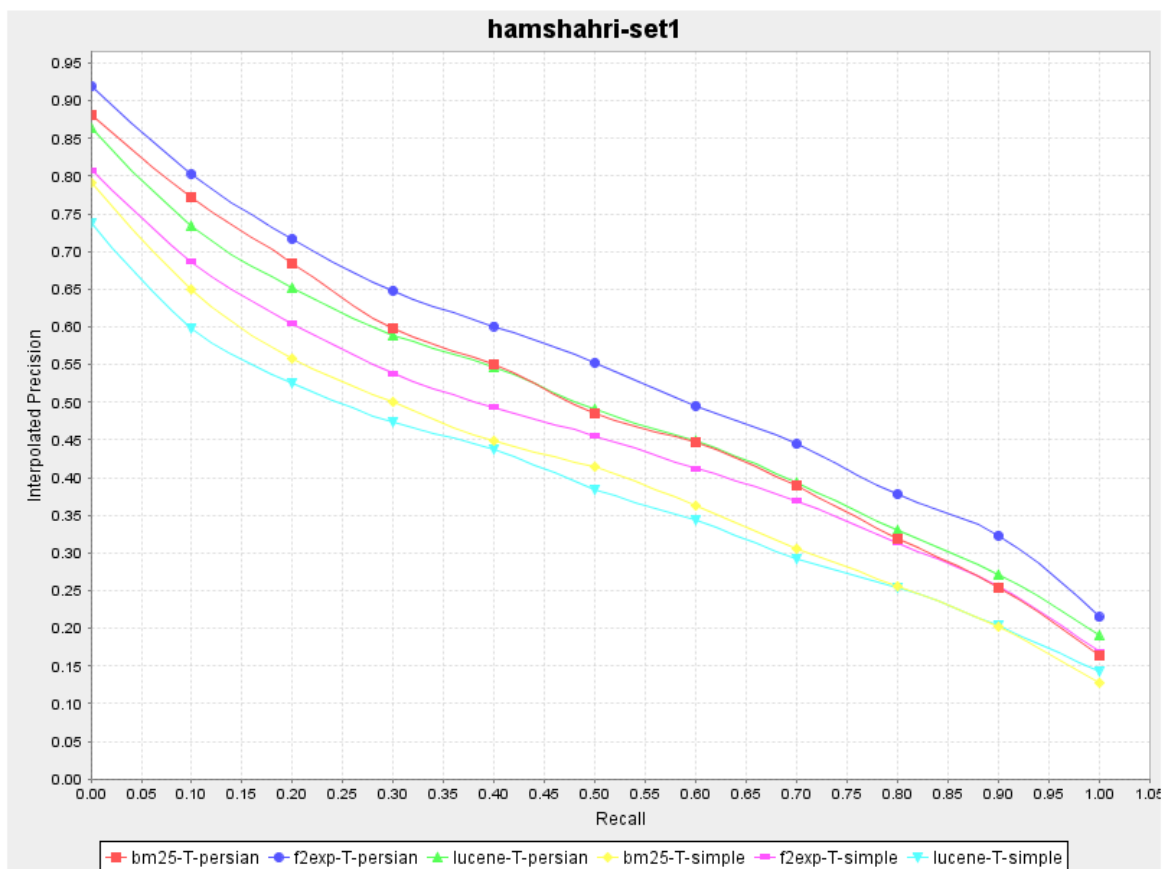


Figura 6.7: Rezultate comparative pe corpusul *Hamshahri* (fără analizor vs. analizor persan)

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-english	2	4967	2067192	1164552	794587	0.6674	0.5464	0.6562	0.9300	0.8839	0.8284	0.6778	0.1600
bm25-T-simple	2	4967	2013945	1164552	816278	0.7035	0.5924	0.6847	0.9455	0.9076	0.8578	0.7036	0.1643
f2exp-T-english	1	4967	2065546	1164552	824684	0.7052	0.5799	0.6881	0.9295	0.8778	0.8320	0.6951	0.1660
f2exp-T-simple	1	4967	2010658	1164552	853600	0.7535	0.6392	0.7300	0.9457	0.9069	0.8648	0.7281	0.1719
lucene-T-english	6	4967	2065546	1164552	818043	0.6930	0.5677	0.6744	0.9294	0.8768	0.8266	0.6872	0.1647
lucene-T-simple	1	4967	2010658	1164552	846109	0.7380	0.6236	0.7131	0.9415	0.9034	0.8586	0.7176	0.1703

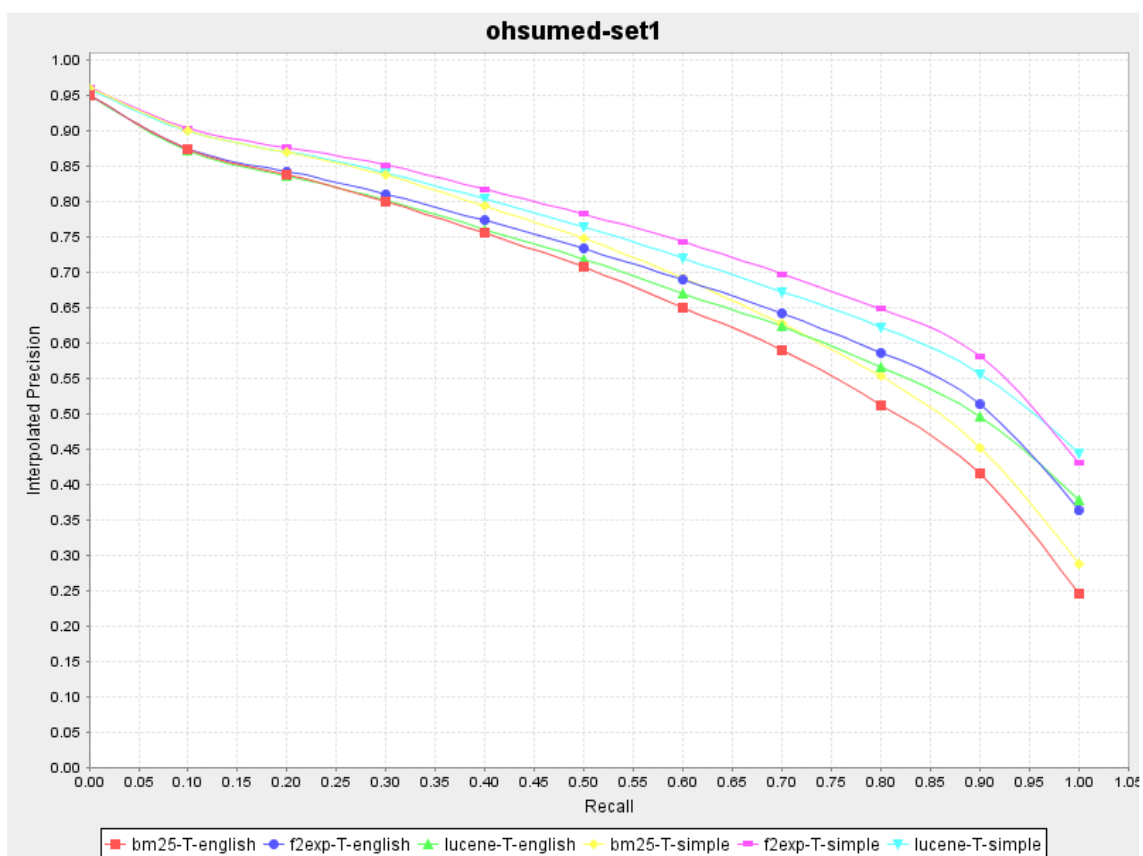


Figura 6.8: Rezultate comparative pe corpusul *Ohsumed* (fără analizor vs. analizor englez)

6.3.3 Cluster pruning

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-simple	2	4967	2013945	1164552	816278	0.7035	0.5924	0.6847	0.9455	0.9076	0.8578	0.7036	0.1643
clusterpruning-T-simple	1	4965	1727971	1164205	405676	0.3460	0.0404	0.3438	0.7953	0.6870	0.5473	0.3989	0.0817
f2exp-T-simple	1	4967	2010658	1164552	853600	0.7535	0.6392	0.7300	0.9457	0.9069	0.8648	0.7281	0.1719
lucene-T-simple	1	4967	2010658	1164552	846109	0.7380	0.6236	0.7131	0.9415	0.9034	0.8586	0.7176	0.1703

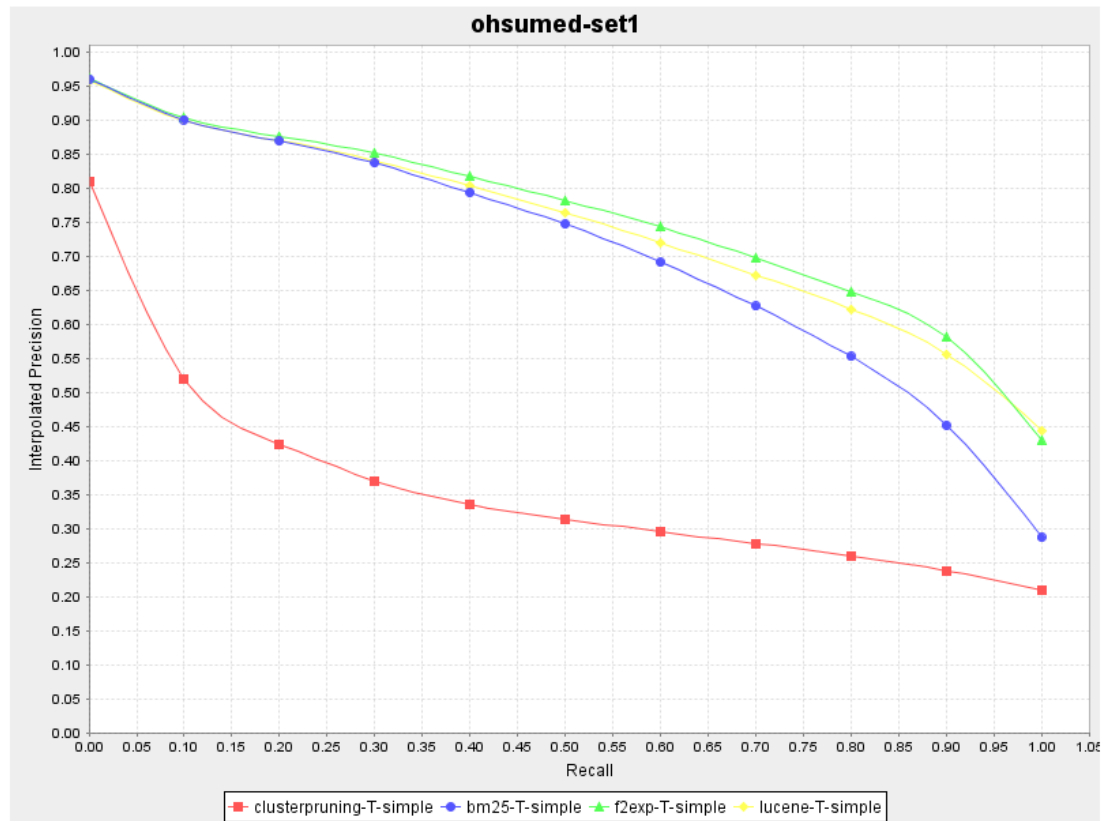


Figura 6.9: Performanțe slabe *cluster pruning* comparativ cu restul metodelor (nici viteza nu excelează)

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
clusterpruning-T-english	1	3979	1503092	943951	157589	0.1482	0.0011	0.1589	0.3822	0.3045	0.2511	0.1879	0.0396
clusterpruning-T-simple	1	4965	1727971	1164205	405676	0.3460	0.0404	0.3438	0.7953	0.6870	0.5473	0.3989	0.0817
lucene-T-simple	1	4967	2010658	1164552	846109	0.7380	0.6236	0.7131	0.9415	0.9034	0.8586	0.7176	0.1703

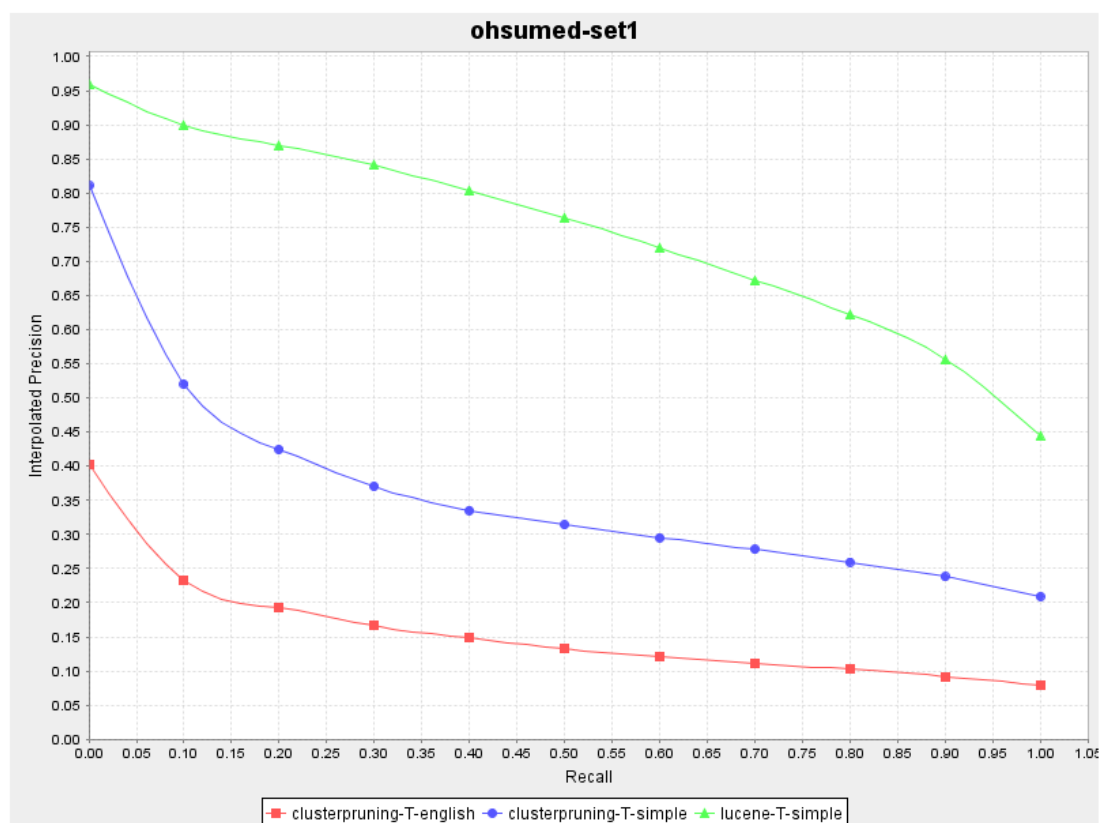


Figura 6.10: Comportament mai slab în cazul folosirii listelor de stop și stemmer-ului englez pentru *Cluster pruning*

6.3.4 Îmbunătățirea performanțelor

Inițial, BM25 a avut rezultate destul de slabe față de metoda implicită lucene bazată pe VSM. Pentru a rectifica acest lucru am făcut două ajustări.

1. ajustarea formulei *idf* astfel încât să se evite rezultatele negative și denaturarea scorului adunând 1 la cantitatea logaritmată (vezi secțiunea 4.3.3 și figura 6.12 pentru rezultate); această anomalie apare doar la interogările mai lungi, care au șanse mai mari să conțină termeni care apar în multe documente;
2. folosirea componentei *coord()* descrisă în secțiunea care tratează modul de obținere a scorului de similaritate folosit de *Lucene*; această componentă aduce o creștere destul de mare a calității rezultatelor așa cum se observă în figura 6.11.

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
#idf+1#bm25-T-simple	2	4967	2013945	1164552	693665	0.5645	0.4341	0.5664	0.9418	0.8940	0.8013	0.6159	0.1397
#idf+1,coord#bm25-T-simple	2	4967	2013945	1164552	816278	0.7035	0.5924	0.6847	0.9455	0.9076	0.8578	0.7036	0.1643
bm25-T-simple	2	4967	2013945	1164552	692370	0.5630	0.4299	0.5653	0.9399	0.8914	0.7991	0.6144	0.1394

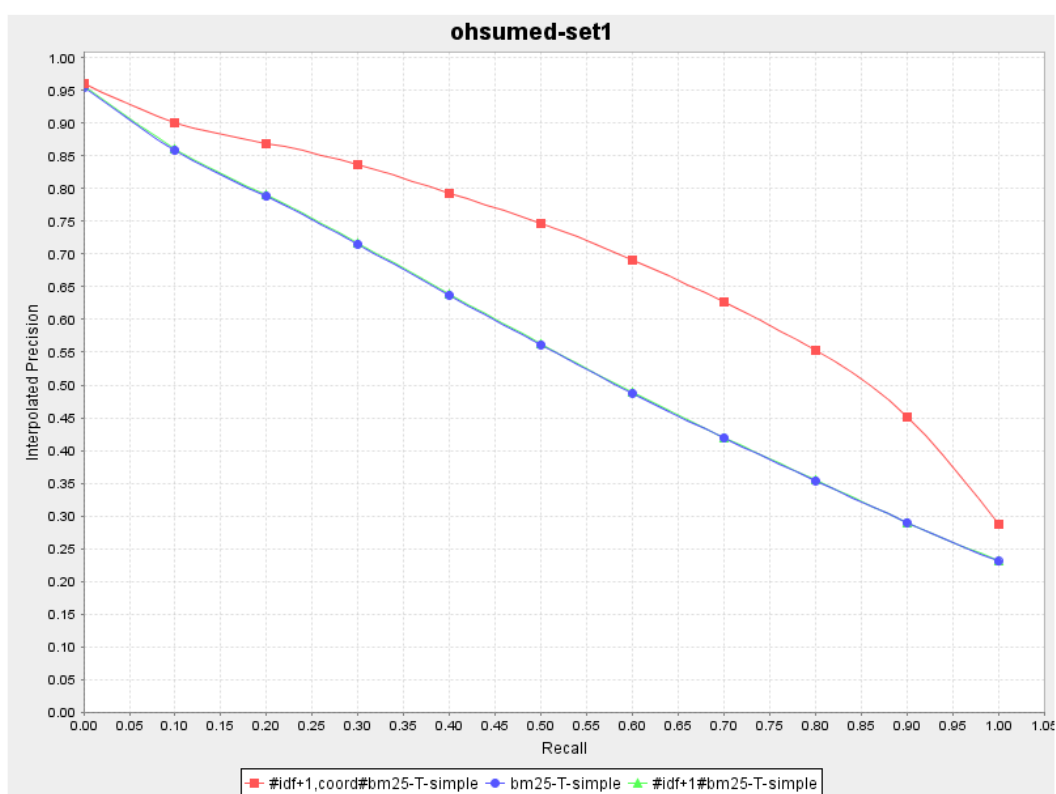


Figura 6.11: Creșterea substanțială a performanțelor pentru *bm25* la folosirea normalizării *coord()*(vezi secțiunea 6.1.1)

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
#idf+1#bm25-T-simple	37	63	31500	3205	1026	0.0595	0.0043	0.0803	0.2014	0.1206	0.0825	0.0652	0.0163
bm25-T-simple	38	63	31500	3205	958	0.0541	0.0032	0.0768	0.1965	0.1111	0.0767	0.0573	0.0152

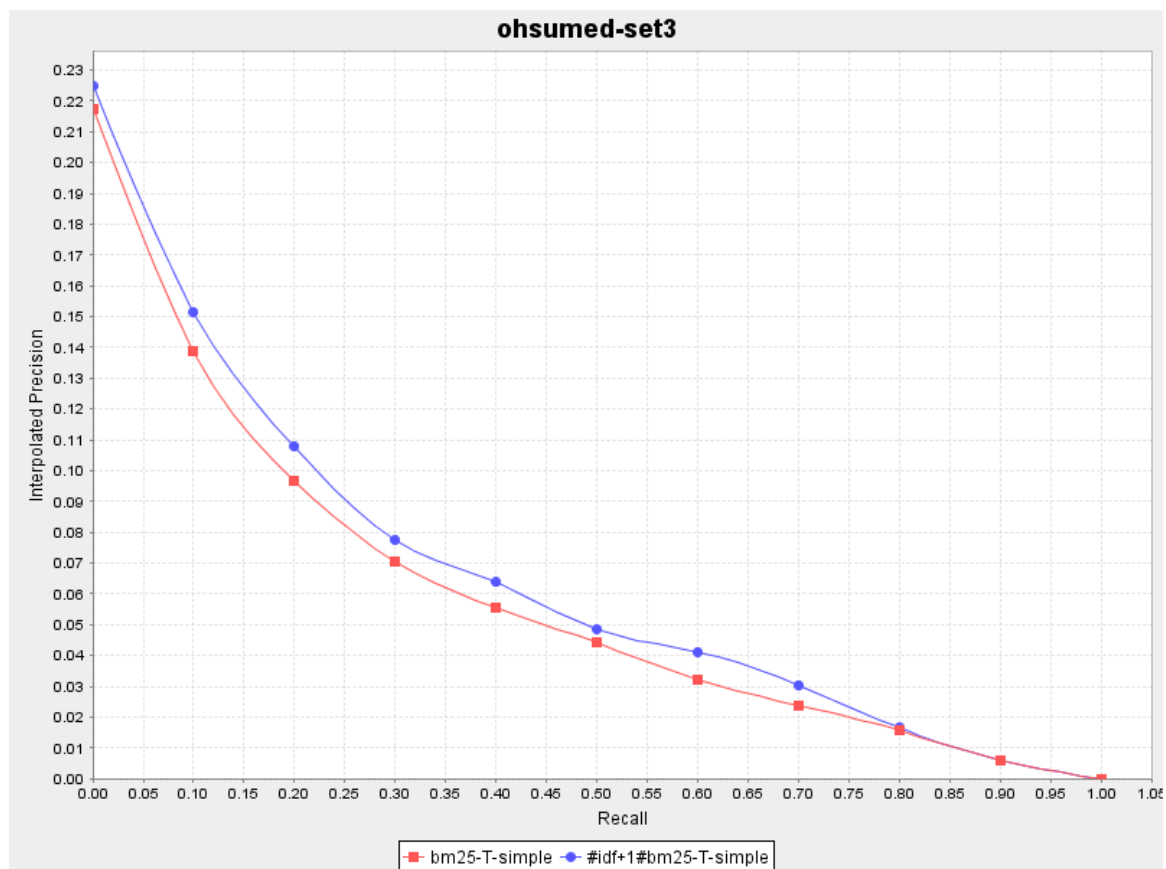


Figura 6.12: Creșterea performanțelor pentru *bm25* la modificarea formulei *idf* pentru evitarea rezultatelor negative din cauza logaritmării unui număr subunitar

6.3.5 Agregare

Metodele de agregare nu au avut rezultate notabile pe cele trei corpusuri.

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-simple	8	35	17411	2344	2036	0.5109	0.4623	0.5091	0.7938	0.6171	0.4971	0.3354	0.0582
f2exp-T-simple	5	35	17411	2344	2020	0.5023	0.4498	0.4963	0.8414	0.6000	0.4933	0.3266	0.0577
lucene,bm25,f2exp-borda-T-simple	35	35	17411	2344	2035	0.5051	0.4548	0.5018	0.8095	0.6114	0.4905	0.3320	0.0581
lucene,bm25,f2exp-rda-T-simple	11	35	17411	2344	2030	0.5042	0.4540	0.4988	0.8167	0.6000	0.4857	0.3323	0.0580
lucene-T-simple	4	35	17411	2344	2031	0.5033	0.4542	0.5037	0.8167	0.6000	0.4886	0.3311	0.0580

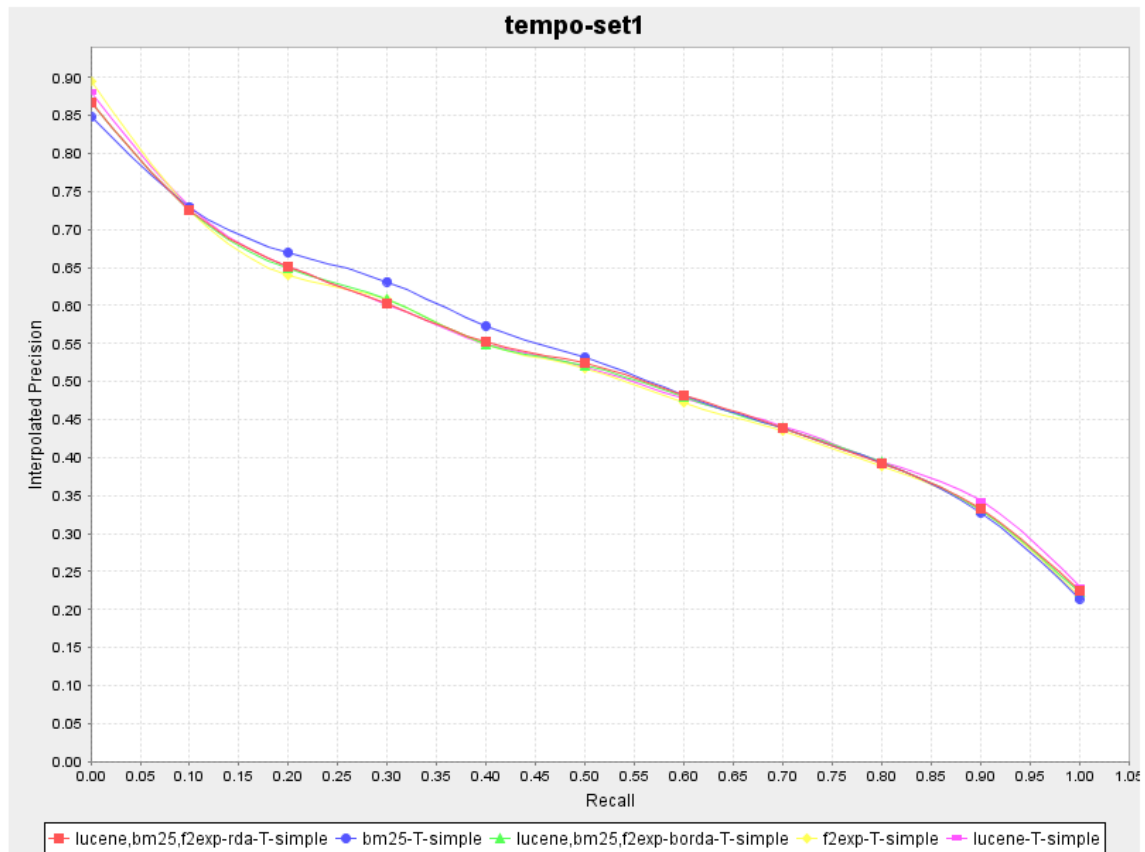


Figura 6.13: Rezultate agregare *Borda* și *RDA* pe corpusul *Tempo*

Metode	QT(ms)	Q	RET	REL	REL+RET	MAP	MGP	R-PR	MRR	P5	P30	P100	P1000
bm25-T-simple	37	63	31500	3205	1201	0.0730	0.0060	0.0934	0.2532	0.1302	0.1016	0.0784	0.0191
f2exp-T-simple	19	63	31025	3205	1259	0.0818	0.0069	0.1077	0.2226	0.1302	0.1132	0.0889	0.0200
lucene,bm25,f2exp-borda-T-simple	69	63	31500	3205	1304	0.0800	0.0083	0.1048	0.2528	0.1270	0.1095	0.0856	0.0207
lucene,bm25,f2exp-rda-T-simple	67403	63	31500	3205	1288	0.0792	0.0082	0.1001	0.2391	0.1302	0.1079	0.0824	0.0204
lucene-T-simple	17	63	31025	3205	1261	0.0799	0.0063	0.1039	0.2347	0.1238	0.1021	0.0797	0.0200

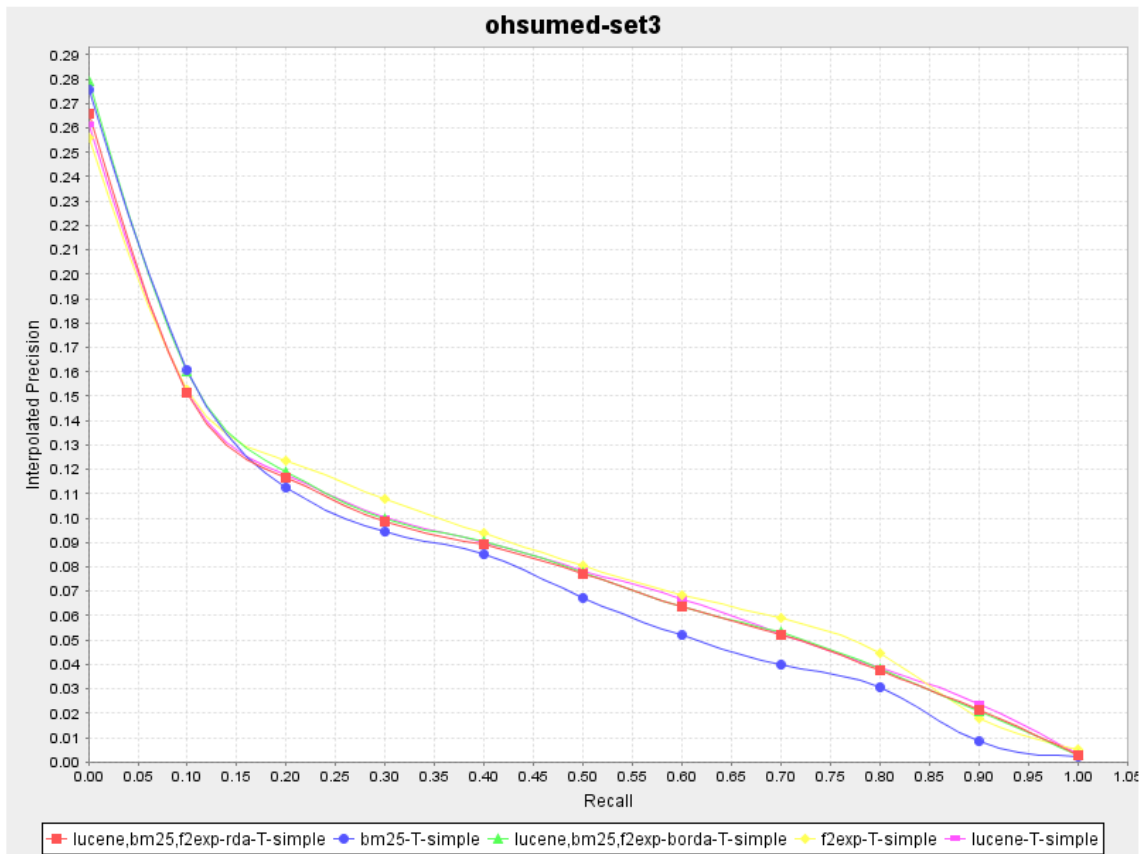


Figura 6.14: Rezultate agregare *Borda* și *RDA* pe corpusul *Ohsumed*, setul de interogări *OHSU*

Bibliografie

- [1] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008.
- [2] David A. Grossman, Ophir Frieder , *Information retrieval: algorithms and heuristics, 2nd Edition*
- [3] <http://www.search-engines-book.com/>
- [4] http://en.wikipedia.org/wiki/Information_retrieval
- [5] [http://en.wikipedia.org/wiki/Index_\(search_engine\)](http://en.wikipedia.org/wiki/Index_(search_engine))
- [6] http://en.wikipedia.org/wiki/Inverted_index
- [7] Hui Fang, ChengXiang Zhai, *An Exploration of Axiomatic Approaches to Information Retrieval*
- [8] Liviu P. Dinu, Florin Manea, *An Efficient Approach for the Rank Aggregation Problem*
- [9] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, *Rank aggregation methods for the Web*, Proc. of the 10th International WWW Conference, 2001.
- [10] <http://lucene.apache.org/java/docs/>
- [11] http://lucene.apache.org/java/3_0_1/api/core/index.html
- [12] <https://cwiki.apache.org/ORP/>
- [13] http://trec.nist.gov/trec_eval/