SISTEM DE CĂUTARE PERSONALIZATĂ PENTRU REORDONAREA REZULTATELOR DIN CĂUTĂRILE WEB

Ciprian Cândea

Ropardo SRL, Sibiu - Departamentul de Cercetare şi Dezvoltare

ciprian.candea@ropardo.ro

Rezumat: Acest articol prezintă un sistem de căutare personalizată pentru reordonarea rezultatelor căutărilor de pagini web. Sistemul este accesibil din browser-ul web. Evaluând paginile web, utilizatorul poate crea profile multiple, care pot fi folosite pentru personalizarea rezultatelor căutărilor furnizate de motorul de căutare. Un algoritm User Profile este folosit pentru a învăța preferințele utilizatorului, folosind un model ierarhic de cuvinte cheie ponderate. Potențialul unui astfel de sistem personalizat este evaluat prin experimente. În plus, doi algoritmi de Information Retrieval, LSI şi FCRN, sunt puşi în aplicare cu scopul de a evalua îmbunătățirile posibile ale procesului de recuperare, folosind datele oferite de User Profile. Un algoritm hibrid, între LSI şi FCRN, este, de asemenea, propus şi evaluat. Prin experimente, care folosesc performanța motorului de căutare ca o linie de referință, sunt prezentate beneficiile sistemului de căutare personalizat.

Cuvinte cheie: căutare personalizată, profilul utilizatorului, regăsirea informațiilor, metrici de similaritate, Singular Value Decomposition, evaluare.

Abstract: This paper presents a personalized search system for re-ranking web search results. The system is accessible from the web browser. By rating web pages, the user may create multiple profiles that can be used to personalize the search results provided by the search engine.

A User Profile algorithm is used to learn the user's preferences using a hierarchical weighted keywords model. The potential of such personalized system is evaluated through experiments. Furthermore, two Information Retrieval algorithms, LSI and FCRN, are implemented in order to evaluate potential improvements of the retrieval process, using data from the User Profile. A hybrid algorithm, between LSI and FCRN, is also proposed and evaluated.

Through experiments, which use the performance of the search engine as a baseline, the benefits of the personalized search system are presented.

Keywords: Personalized Search, User Profile, Information Retrieval, Similarity Metrics, Singular Value Decomposition, Evaluation.

1. Introducere

Un sistem de căutare personalizată [1] este definit ca un tip de sistem de Information Retrieval (IR) [2], care personalizează căutarea de informații, de la un utilizator la altul. Prin calcule contextuale, pentru orice utilizator, mediul de calcul se adaptează la fiecare punct de calcul. Preferințele şi comportamentele utilizatorilor sunt modelate. Una dintre cele mai importante caracteristici ale unui sistem de căutare personalizat este că relevanța este relativă pentru fiecare utilizator. În loc de a avea o relevanță a informației identică pentru toți utilizatorii (determinată eventual de o mulțime de experți), căutarea personalizată propune contextualizarea şi individualizarea relevanței informației.

Există două tehnici generale utilizate de sisteme de căutare personalizate: (1) reordonarea rezultatelor de căutare şi (2) augmentarea interogării şi procesarea rezultatului [1]. Prima tehnică presupune reordonarea rezultatelor obținute de către utilizator, după o operațiune de căutare, prin măsurarea similarității rezultatelor cu preferințele utilizatorului, stocate într-un profil de utilizator. A doua abordare încearcă să rafineze o interogare (de căutare) a utilizatorului prin compararea termenilor din interogare cu informațiile individuale şi contextuale asociate cu acel utilizator. Astfel, sistemul ar trebui să fie capabil de a propune o interogare mai relevantă (din perspectiva utilizatorului respectiv).

Sisteme de căutare personalizate sunt cel mai frecvent utilizate în serviciile de căutare web. Google [3], Bing [4] şi Yahoo [5] au introdus astfel de sisteme. Deşi puține detalii sunt cunoscute despre sistemele lor (proprietare), este de aşteptat ca limba utilizatorului, locația şi istoricul web să fie factorii cheie adoptați [6]. De exemplu, căutarea personalizată de la Google funcționează cel mai bine în cazul în care utilizatorul are un cont cu istoricul web activ. Totuşi, sistemul lor poate lucra şi cu utilizatorii neautentificați printr-un cookie în browser-ul web.

În această lucrare propunem un sistem de căutare personalizată pentru reordonarea rezultatelor căutărilor web pe care l-am conceput şi dezvoltat folosind bine-cunoscuta arhitectură software de tipul client-server. În timp ce serverul este responsabil cu menținerea profilelor pentru mai mulți utilizatori, clientul este disponibil ca plug-in pentru browser-ul web Mozilla Firefox. Datele sunt schimbate între clienți şi server prin intermediul serviciilor web. Algoritmul pentru modelarea profilului de utilizator din [7] a fost implementat. Abordarea propusă este detaliată în secțiunea 3, după o imagine de ansamblu a cercetărilor similare, care este prezentată în secțiunea 2. Algoritmul de profil al utilizatorului este, de asemenea, combinat cu alți algoritmi IR, cu scopul de a încerca să se îmbunătățească performanțele sale. Obiectivul principal al acestei lucrări este de a evalua şi a compara mai mulți algoritmi care pot fi utilizați pentru un sistem de căutare personalizată. După prezentarea metodologiei noastre de evaluare (a se vedea secțiunea 4), se prezintă mai multe experimente, în secțiunea 5, cu scopul de a demonstra avantajele unei abordări de tipul căutare personalizată, precum şi pentru a arăta modul în care algoritmii adoptați performează, folosind motorul de căutare Google ca referință. În cele din urmă, secțiunea 6 prezintă concluziile şi direcțiile noastre pentru activitatea viitoare.

2. Cercetări similare

Trei tipuri principale de sisteme de căutare web personalizate sunt identificate în [8]: (1) pe bază de hyperlink, (2) site-uri web personalizate şi (3) sisteme de recomandare.

Sistemele de căutare bazate pe hyperlink propun utilizarea unui PageRank personalizat, în loc de unul global. PageRank este în esență calculat folosind conexiunile (link-urile) dintre paginile web.

Site-urile web personalizate se orientează pe personalizarea link-urilor şi a conținutului. De exemplu, site-uri web de e-commerce personalizează conținutul web în funcție de preferințele utilizatorilor.

Sistemele de recomandare încearcă să ajute utilizatorii să navigheze prin tot mai multele informații disponibile on-line. Recomandarea pe bază de filtrare colaborativă şi cea bazată pe conținut sunt cele două abordări principale.

Diferite tipuri de activități ale utilizatorilor (de exemplu, pagini web vizitate anterior şi interogări) sunt înregistrate pe o perioadă lungă de timp, în scopul de a construi un profil de utilizator implicit [9]. Apoi, ele sunt analizate în mod automat pentru a oferi un feedback relevant pentru un sistem de căutare web personalizat. O altă abordare bazată pe un profil de utilizator generat automat, propune un sistem personalizat de căutare web bazat pe o ontologie [10]. O ierarhie de concepte este utilizată pentru a caracteriza paginile web accesate de utilizatori.

În [11] se arată că stabilirea în mod automat a intereselor utilizatorilor se poate face chiar din date ale istoricului de click-uri. Datele acestea sunt, de asemenea, utilizate într-un alt sistem personalizat de căutare web, care este dezvoltat ca un sistem de recomandare [12]. CubeSVD este un tip de Singular Value descompunere (SVD), utilizat pentru a analiza datele istoricului de click-uri, reprezentate de trei caracteristici: uşurința în utilizare, interogare şi pagină web.

UCAIR este un agent de căutare web [13] construit pe baza motorului de căutare Google care, pe lângă reordonarea paginilor web preluate, efectuează, de asemenea, augmentarea interogărilor. Feedback-ul de la utilizator este implicit şi este obținut în principal din istoricul de click-uri şi interogările din trecut. Cantitatea datelor istorice examinate este cercetată şi în [14], [8].

În loc de a considera astfel de date istorice, în această cercetare, am ales să se folosească feedback-ul explicit de la utilizator (în scopul de a construi un profil). Acest feedback informează sistemul cât de mult utilizatorul a apreciat sau nu o anumită pagină web. Abordarea noastră nu se limitează la această tehnică de profil al utilizatorului. Cu toate acestea, am ales o astfel de abordare inițială (mai uşor de pus în aplicare), pentru că suntem în principal interesați în evaluarea performanțelor unui sistem de căutare personalizată şi nu (acum) de optimizarea unui astfel de sistem. (Este cunoscut faptul că un sistem de căutare personalizată este afectat de efectul Bubble Filter [15].) Cu astfel de feedback, scopul nostru este de a crea un sistem de recomandare pe baza conținutului. La fel ca sistemul UCAIR, am dezvoltat un agent de căutare web construit pe baza

Google. Sistemul nu se limitează la acest motor de căutare şi este, de asemenea, capabil de prelucrarea paginilor web scrise în limbi diferite. În plus, ca şi în [12], folosim algoritmi bazaţi pe SVD.

3. Abordarea Propusă

În această secţiune este prezentată metoda propusă pentru un sistem de căutare personalizată. Această metodă (a se vedea Figura 1) poate fi aplicată pe orice colecţie de documente care pot fi indexate. Astfel de documente pot fi documente medicale, articole de cercetare, cărţi, pagini web etc.

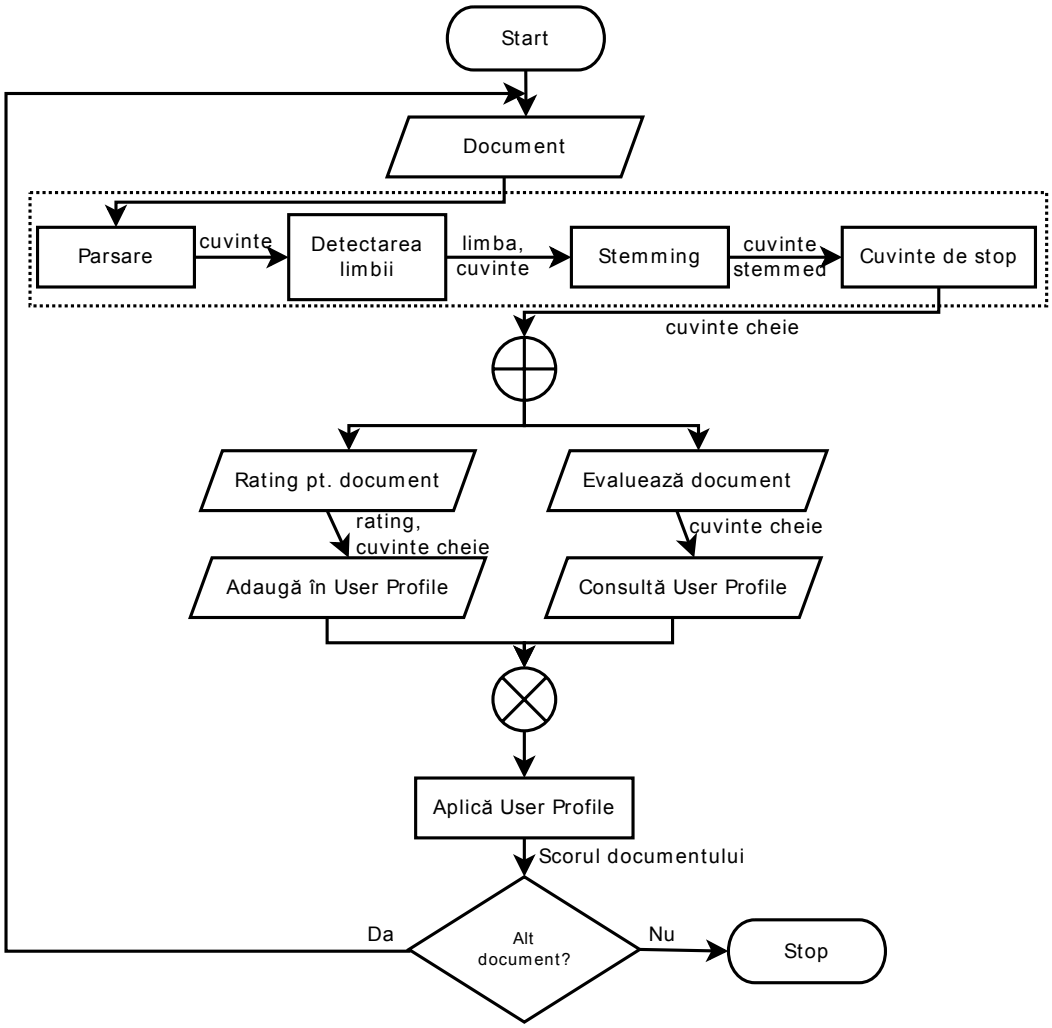

Figura 1. Metoda propusă pentru profilul utilizatorului

Iniţial, profilul utilizatorului este gol. Procesul din Figura 1 începe atunci când un utilizator introduce document. Documentul este apoi prelucrat folosind tehnici de procesare a limbajului natural (NLP – Natural Language Processing): (1) parsare, (2) detectarea limbii, (3) stemming şi (4) îndepărtarea cuvintelor de stop.

Este important de menţionat că, în timp ce ne bazăm pe (bine cunoscutele) metode de NLP de mai sus, abordarea noastră se aplică în domeniul de cercetare Information Retrieval (IR). Acest lucru se datorează faptului că suntem în principal axaţi pe recuperarea, căutarea şi stocarea informaţiilor.

Etapa de parsare ia ca intrare documentul şi generează cuvintele din el. Apoi, aceste cuvinte sunt folosite ca şi conţinut al profilului, care este comparat cu profiluri lingvistice generice bazate pe materiale din diferite surse (de exemplu: [16]). Pentru detectarea limbii, modelul n-gram (un

model Markov de ordinul *(n-1)*) este utilizat [17], cu $n = 3$.

După ce este identificată limba documentului, se aplică un algoritm de Stemming corespunzător. Stemmer-ul Porter [18] este utilizat pentru limba engleză. Alte limbi, cum ar fi italiană, română, franceză, utilizează stemmer-e Romance. Limba germană are propriul său Stemmer.

Cuvintele tulpină (stemmed) sunt apoi filtrate cu ajutorul unor liste (specific lingvistice) de cuvinte de stop. În cele din urmă, o listă de cuvinte cheie este obținută din documentul utilizatorului.

Utilizatorul are apoi două opțiuni. Documentul poate fi comparat cu profilul curent pentru a obține un scor pentru el. Evident, acest scor va indica relevanța documentului în profilul utilizatorului. Acesta poate fi folosit pentru a clasifica mai multe documente. Sau, utilizatorul poate da un rating la document şi apoi îl poate introduce în profilul său.

În ambele cazuri, un algoritm User Profile trebuie să fie aplicat. Un astfel de algoritm ia ca intrare cuvintele cheie şi furnizează un scor. În cazul adăugării documentului în profilul utilizatorului, este de asemenea necesar feedback-ul pentru acest document. Desigur, utilizatorul poate repeta acest proces. Astfel, profilul de utilizator va evolua în timp. Evident, algoritmul User Profile poate varia în funcție de tipul de documente.

În această lucrare, abordarea de mai sus este utilizată cu scopul de a îmbunătăți calitatea căutărilor web.

Figura 2 rezumă această tehnică.

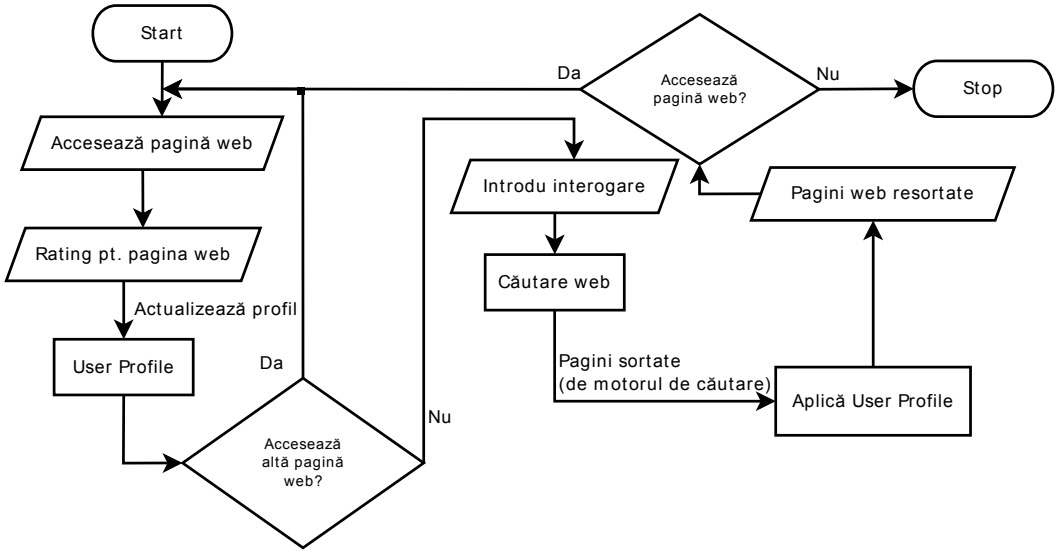

Figura 2. Algoritmul User Profile pentru ordonarea paginilor web

Dintr-un browser web, un utilizator accesează o pagină web, având posibilitatea de a-i da un rating. Orice pagină web care a primit rating va intra în profilul utilizatorului. În cazul în care utilizatorul va căuta pe web, un motor de căutare cunoscut (de ex.: Google) va oferi mai multe pagini web, într-o anumită ordine. Cu toate acestea, acest rezultat ar putea să nu fie cel mai bun pentru preferințele utilizatorului. În acest moment, paginile web sunt marcate cu ajutorul profilului de utilizator. Ele sunt apoi re-sortate şi prezentate utilizatorului.

3.1 Algoritmul User Profile

Algoritmul User Profile implementat permite modelarea profilului de utilizator pe baza unui feedback explicit. Acest feedback constă în cât de mult îi place utilizatorului un document. Cercetarea noastră este axată pe profiluri de utilizator construite din pagini web de pe Internet. Un motor de căutare recomandă pagini web pentru utilizator pe baza unei interogări date. În cazul în

care utilizatorul accesează o pagină web, el poate evalua acea pagină. Am ales o schemă de rating din cinci stele, cu următoarea semnificație: o stea → α = -0.7, 2 stele → α = -0.3, 3 stele → α = 0,1, 4 stele → α = 0,3 şi 5 stele → α = 0,7. α este feedback-ul (de asemenea, numit rata de învățare) pentru algoritmul User Profile. Valoarea sa creşte treptat, de la un feedback negativ (o stea) la un feedback pozitiv (cinci stele). Rețineți că semnul minus este folosit doar pentru a marca un feedback negativ. Altfel, algoritmul utilizează valoarea α absolută. Profilul utilizatorului este construit din una sau mai multe categorii, care pot varia în timp. Numărul maxim de categorii este setat la 100, dar este parametrizabil. După cum se arată în Figura 3, fiecare categorie are trei descriptori: pozitiv, pe termen lung şi negativ.

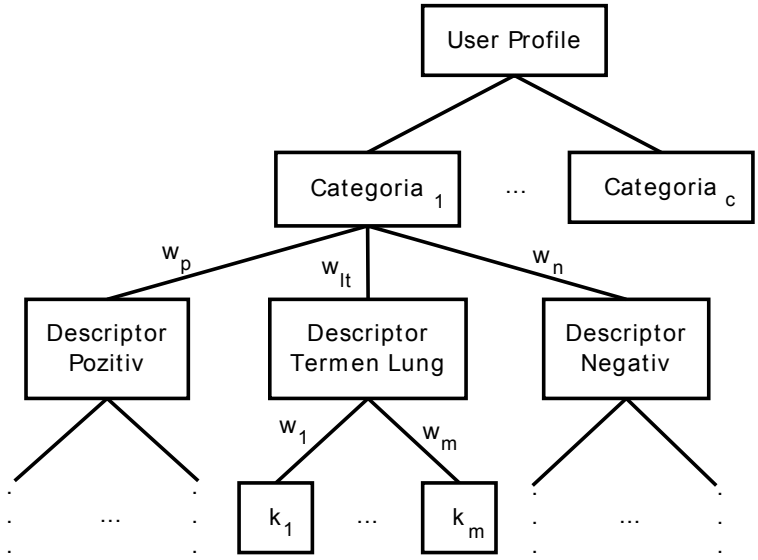

Figura 3. Modelul User Profile

Fiecare descriptor reține cuvinte cheie ponderate. Un maxim de 100 de cuvinte cheie este permis în mod implicit, dar acest lucru este parametrizabil. Descriptorii pozitiv şi negativ sunt formați din documente cu feedback-ul pozitiv şi respectiv, negativ. Ei acționează pe termen scurt. Descriptorul pe termen lung conține cuvinte cheie din documente cu orice fel de feedback. Fiecare descriptor are o pondere globală, adică un nivel de interes pentru categoria pe care o reprezintă.

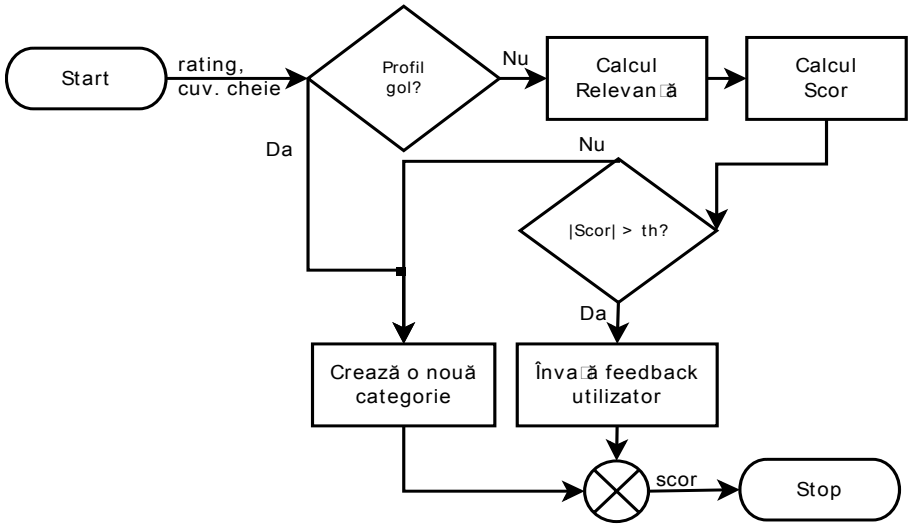

Figura 4. Algoritmul User Profile

Algoritmul (Figura 4) porneşte de la un profil de utilizator gol. Primul vector caracteristic (o serie de cuvinte cheie extrase dintr-un document) pe care îl primeşte este folosit pentru a crea

prima categorie a profilului. Cuvintele cheie din acest vector sunt utilizate pentru a calcula ponderile celor trei descriptori din noua categorie. Feedback-ul (`alfa`), dat de utilizator influențează modul în care ponderile sunt calculate. De exemplu, descriptorul negativ pe termen scurt (`STN`) va avea greutatea calculată numai în cazul în care feedback-ul este negativ. Atunci când profilul de utilizator nu este gol, algoritmul calculează relevanța fiecărei categorii din profil în raport cu noul vector de caracteristici. Categoria cea mai relevantă este selectată cu relația:

$$Relevanța(C, fv) = \max\{sim(d_p^C, fv), sim(d_{lt}^C, fv), sim(d_n^C, fv)\}. \quad (1)$$

Notăm cu *C* categoria din profil pentru care se calculează relevanța. *fv* (feature vector) este un vector caracteristic. Este reprezentarea vectorială a unei pagini web. Pentru fiecare dintre cei trei descriptori (*d*) o metrică similaritate (*cosinus*) este calculată în raport cu vectorul caracteristic. Categoria cu cea mai mare similaritate (pe oricare dintre descriptori) este considerată cea mai relevantă.

Apoi, scorul a unei pagini web (raportat la categoria cea mai relevantă) se calculează folosind relația următoare:

$$Scor(fv) = \max\{w_{lt}, w_p\} + \min\{w_{lt}, -w_n\}. \quad (2)$$

Ponderea descriptorului pozitiv este considerată ori de câte ori ponderea descriptorului pe termen lung este mai mică. În mod similar, ponderea descriptorului negativ este considerată ori de câte ori ponderea descriptorului pe termen lung este mai mare. Rețineți că $w_p, w_n \in [0,1]$ și $w_{lt} \in [-1,1]$. Acest lucru justifică prezența semnului minus pentru w_n.

Dacă acest scor, în valoare absolută, este mai mare decât o valoare de prag stabilită de utilizator, vectorul caracteristic va fi introdus în profilul de utilizator. Acesta va fi absorbit în categoria (existentă) cea mai relevantă. În caz contrar, vectorul caracteristic va fi folosit pentru a crea o nouă categorie în profil. Pragul este un număr subunitar. Cu cât pragul este mai mare, mai multe categorii vor fi create în profilul de utilizator, care va fi mai dilatat. Cu cât pragul este mai mic, mai puține categorii vor fi prezente în profilul de utilizator.

Nu insistăm aici asupra modului în care sunt calculate ponderile fiecărui descriptor şi ale cuvintelor cheie. Mai multe detalii despre acest algoritm sunt disponibile în [7]. Totuşi, merită menționat faptul că algoritmul aplică o rată descompunere (0,97 implicit) privind ponderile, după o anumită perioadă de timp (o săptămână, implicit). Aceasta este o abordare simplă pentru mimarea unui proces de îmbătrânire a profilului de utilizator. Ca parametrii de intrare, implementarea noastră a algoritmului foloseşte: identificator de utilizator (pentru asocierea profilului cu o persoană), URL-ul (Uniform Resource Locator pentru pagina de web evaluată), *alfa* (ratingul dat de utilizator pentru pagina web) şi *th* (pragul menționat mai sus). Ca ieşire, algoritmul poate oferi un scor pentru documentul furnizat.

Am implementat acest algoritm User Profile folosind o arhitectură software client-server. O extensie Mozilla Firefox serveşte ca şi client. Profilele de utilizator sunt stocate pe partea de server, într-o bază de date relațională. Clientul comunică cu serverul prin servicii web REST [19]. Orice utilizator se poate înregistra în sistem folosind un e-mail şi o parolă. Utilizatorilor li se permite să aibă mai multe profiluri. Mai mult decât atât, profilele sunt asociate cu o limbă. Limba de pe o pagină web este detectată în mod automat prin recunoaşterea de trigrame. Pentru prelucrarea paginilor web, s-au folosit instrumente precum Snowball (http://snowball.tartarus.org/) şi JavaScript language identifier (https://github.com/mazko/jsli).

3.2 Algoritmi de Information Retrieval

În Information Retrieval [2], documentele şi interogările sunt reprezentate ca vectori în spațiu *t*-dimensional, unde *t* este numărul de termeni indexați în colecție. Latent Semantic Indexing (LSI) [20] este o variantă de Vector Space Model (VSM) [21], în care matricea originală termen-document este descompusă, folosind Singular Value Decomposition (SVD) [22], în trei matrici: U, o matrice termen după dimensiune (m × m), S o matrice valoare singulară (dimensiune după

dimensiune, (m × n)), şi V, o matrice document după dimensiune (n × n). Reţineţi că pentru orice matrice A, există o descompunere SVD. Matricea originală poate fi obţinută, prin multiplicarea matriceală $U*S*V^T = A$ (m×n).

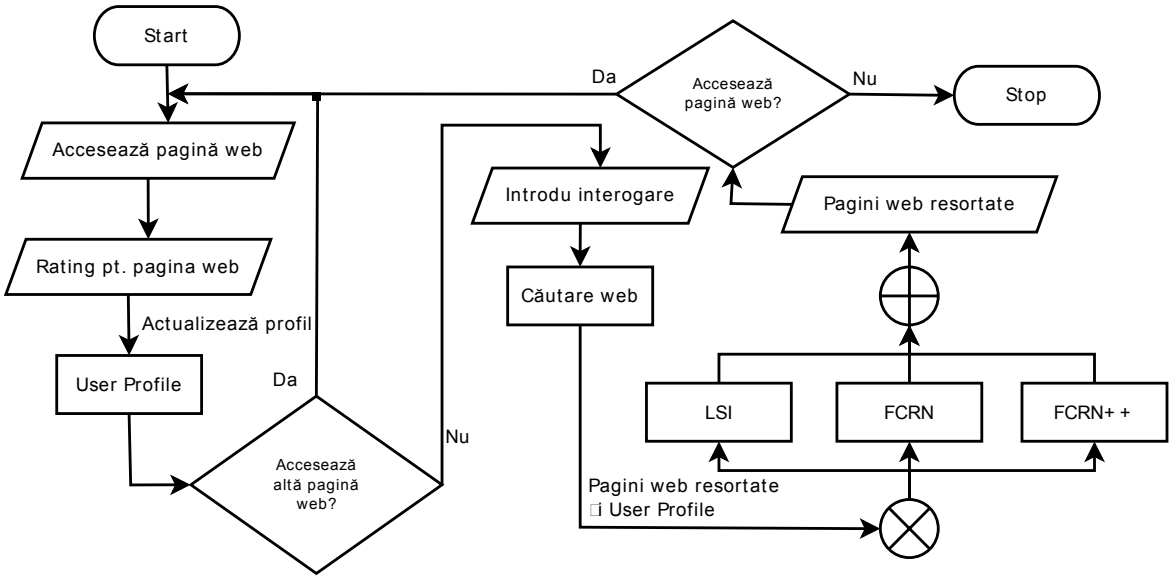

Figura 5. Algoritmi Information Retrieval pentru ordonarea paginilor web

Ne propunem folosirea profilului utilizatorului, ca şi intrare pentru un algoritm Information Retrieval (LSI, FCRN, sau FCRN++), care este capabil de a reordona paginile web obţinute (prin motorul de căutare), astfel că ordinea rezultată este mai adecvată pentru preferinţele utilizatorului. Evident, utilizatorul va avea posibilitatea de a da rating acestor pagini web reordonate. Astfel, profilul de utilizator continuă să evolueze şi să înveţe mai bine preferinţele acestuia.

În esenţă, abordarea noastră constă într-un Sistem de Recomandare de tip Factorizarea unei Matrice [23]. În următoarele secţiuni vom prezenta aceşti algoritmi IR din Figura 5.

În comparaţie cu abordarea din Figura 2, se propune aici completarea algoritmului User Profile cu algoritmi IR. Obiectivul este o comparaţie de performanţă.

3.1.1 Latent Semantic Indexing

Într-un sistem LSI, matricele U, S şi V sunt trunchiate la *k* dimensiuni. Scopul acestei reduceri de dimensiune este de a reduce zgomotul în spaţiul latent, rezultând o structură mai bogată de relaţii între cuvinte care arată semantica latentă prezentă în colecţie. *k*-ul optim este determinat empiric pentru fiecare colecţie. În general, valori ale lui *k* mai mici sunt preferate atunci când se utilizează LSI, datorită costului de calcul asociat cu algoritmul SVD, precum şi costurile de stocare şi comparare a vectorilor de mari dimensiuni. Este prezentată în [22] o teoremă, bazată pe norma Frobenius, care calculează procentul de schimbări relative între matricea A la rang *(k + 1)* şi matricea A la rang *k*. Doar o *p%* schimbare relativă este necesară pentru a reduce gradul matricei A de la *r* la *k*. Cu cât *k* este mai mic, cu atât mai mare este utilizarea spaţiului latent, care se traduce în mai multe relaţii latente să fie găsite de către LSI. Cu cât este mai mare *k*, un număr mai mic de relaţii latente vor fi obţinute. Când *k* este maxim, nicio relaţie latentă nu va fi disponibilă. LSI va acţiona ca VSM în acest caz. Numai cuvinte cheie prezente într-un document vor fi în relaţie cu acel document.

Este important pentru metoda LSI ca matricea A_k derivată să nu reconstruiască exact matricea originală termen document (A). SVD trunchiat, într-un sens, surprinde cele mai importante părţi din structura de bază a asocierii de termeni şi documente dar, în acelaşi timp, elimină zgomotul sau variabilitatea în utilizarea cuvintelor care afectează metodele de recuperare bazate pe cuvinte.

Algoritmul LSI se bazează pe descompunerea SVD a matricei A. Acest lucru permite documentelor, termenilor şi interogărilor să fie reprezentate în spaţiu *k*-dimensional [22]:

- coordonatele documentelor în spaţiul *k*-dimensional sunt date de liniile matricei V_k.;
- coordonatele interogării în spaţiul *k*-dimensional sunt date de următoarea relaţie:

$$q = q^T * U_k * S_k^{-1} \quad (3)$$

Vectorul interogare (q) poate fi apoi comparat cu toţi vectorii de documente existenţi şi documentele pot fi clasate prin similaritatea lor cu interogarea. O măsură comună de similaritate este cosinusul dintre vectorul interogare (q) şi vectorul document (d_j):

$$\cos \alpha_j = \frac{q^T \cdot d_j}{\|q\| \cdot \|d_j\|} = \frac{\sum_{i=1}^{m} q_i \cdot a_{ij}}{\sqrt{\sum_{i=1}^{m} q_i^2} \cdot \sqrt{\sum_{i=1}^{m} a_{ij}^2}}$$

Cele mai apropiate *n* documente sau toate cele care depăşesc un anumit prag (de exemplu, *th = 0.8* pentru *k = 2*) sunt returnate utilizatorului.

Implementarea noastră LSI, în Octave / Matlab, este prezentată în Figura 6.

Pas	Algoritmul LSI	Explicaţii
1	`for i = 1 : size(A,2)` `  A(:,i) = A(:,i) /` `  norm(A(:,i));` `end`	*i* ia valori de la 1 la mărimea celei de-a doua dimensiuni a matricei *A* (adică *n*, numărul de coloane). *A(:,i)* este coloana *i* a matricei *A*. *norm($\vec{v}$)* este norma (lungimea) vectorului $\vec{v}$.
2	`[U S V] = svd(A);`	Matricei *A* i se aplică SVD.
3	`Uk = U(:,1:k);` `Sk = S(1:k,1:k);` `Vk = V(:,1:k);` `qv = (q/norm(q))'*Uk*inv(Sk);`	Matricea *Uk* reprezintă primele *k* coloane ale matricei *U*. *Sk* este matricea pătratică de ordin *k* şi conţine primele *k* linii şi primele *k* coloane ale matricei *S*. Matricea *Vk* reprezintă primele *k* coloane ale matricei *V*. Liniile matricei *Vk* reprezintă coordonatele documentelor în spaţiu *k*-dimensional. Coordonatele vectorului interogare (*qv*) în spaţiu *k*-dimensional sunt şi ele calculate. *A'* ($= A^T$) este transpusa matricei *A*. *inv(Sk)* este Sk^{-1}, adică inversa matricei *Sk*.
4	`for j = 1 : size(Vk,1)` `  result(j) =` `  sum(qv.*Vk(j,:)) /` `  (norm(qv)*norm(Vk(j,:)))` `;` `end`	*j* variază de la 1 la lungimea primei dimensiuni a matricei *Vk* (adică *m*, numărul de linii) Este calculat cosinusul dintre vectorul interogare (qv) şi vectorul document *j*. *Vk(j,:)* este linia *j* a matricei *Vk*. *sum(qv.*Vk(j,:))* reprezintă produsul scalar dintre vectorii *qv* şi lina *j* a matricei *Vk*.

Figura 6. Implementarea LSI

Ca input, implementarea LSI foloseşte: matricea *A*, vectorul interogare *q*, parametrul *k* şi parametrul *th*. Matricea termen-document, A (m × n), conţine, în primele *c* coloane, ponderile cuvintelor cheie din descriptorul pe termen scurt pozitiv (al algoritmului User Profile). În următoarele *(n - c)* coloane sunt ponderile cuvintelor cheie din primele *(n - c)* pagini web returnate

de motorul de căutare după aplicarea unui interogări. Fiecare dintre cele *c* coloane corespunde unei categorii din profilul de utilizator. Considerăm că, în acest stadiu, un profil de utilizator există. Apoi, utilizatorul efectuează o interogare folosind un motor de căutare. *(n - c)* pagini web sunt returnate. LSI acționează ca un sistem de recomandare. Se corelează profilul de utilizator cu paginile web returnate astfel încât cele preferate de utilizator să ajungă în partea de sus.

Vectorul interogare este un vector coloană (m x 1), care este definit ca:

$$q = (q_{i1})_{i=1..m} = \begin{cases} 1, & \text{dacă cuvântul cheie este în interogare} \\ 0, & \text{altfel} \end{cases} \quad (4)$$

k este parametrul pentru reducerea dimensională. Valoarea optimă pentru *k* este determinată empiric pentru fiecare colecție.

th este parametrul care stabilește un prag la nivelul de similaritate cosinus, care trebuie luat în considerare la alegerea documentelor cele mai relevante. Este un număr subunitar, ales empiric. Ca output, algoritmul LSI generează o listă ordonată de documente (stocate în *result*). Cele mai relevante (determinate folosind similaritatea cosinus) sunt selectate.

3.1.2 Fast Case Retrieval Net

Case Retrieval Net (CRN) a fost propus ca formalism de reprezentare pentru Case-Based Reasoning (CBR) în [24]. În terminologia LSI, entitățile informaționale (IE) din CRN sunt cuvintele cheie şi cazurile sunt documente. CRN foloseşte funcțiile de relevanță şi de similaritate ca să stabilească relații ponderate dintre IE şi cazuri şi respectiv, între IE-uri. Rețineți că funcția de similaritate este o distincție cheie între LSI şi CRN. Algoritmul de calcul FCRN este mai rapid (decât CRN), deoarece agregă funcțiile de relevanță şi de similaritate într-o singură funcție numită relevanță efectivă. Relevanța efectivă se calculează folosind următoarea relație:

$$\Lambda(IE_i, c) = \sum_{j=1}^{n} \sigma(IE_i, IE_j) * \rho(IE_j, c) \quad (5)$$

unde *c* este un caz, IE_i este o entitate informațională (Information Entity) şi *n* este numărul de IE-uri. (Cu σ şi ρ am notat funcțiile de relevanță şi similaritate.)

Implementarea noastră a FCRN, în Octave/Matlab, este prezentată în Figura 7.

Pas	Algoritmul FCRN	Explicații
1	`% Vezi Figura 6, pasul 1.`	
2	`% Vezi Figura 6, pasul 2.`	
3	`Uk = U(:,1:k);` `Sk = S(1:k,1:k);` `Tk = (Uk*Sk)*(Uk*Sk)';`	Matricea *Uk* constă în primele *k* coloane ale matricei *U*. *Sk* este o matrice pătratică de ordin *k* şi este formată din primele *k* linii şi coloane ale matricei *S*. Matricea *Tk* este matricea termen-termen, care prezintă co-apariția între IE-uri (funcția de similaritate).
4	`E = Tk * A;` `qv = zeros([1, size(E, 2)]);` `for i = 1 : size(q)` `  if q(i) == 1` `  qv = qv + E(i, :);` `  end` `end` `result = (qv/norm(qv))';`	Matricea E este matricea relevanță efectivă (funcția Λ). Relevanța efectivă pentru interogarea *q* (notată cu *qv*) se calculează prin însumarea valorilor *E* pentru toate IE-urile prezente în *q*. Apoi, *qv* este normalizat.

Figura 7. Implementarea FCRN

Ca input, implementarea FCRN utilizează aceleaşi surse ca şi LSI, cu aceeaşi semnificaţie. Acest algoritm generează acelaşi tip de output ca şi LSI.

În comparaţie cu algoritmul FCRN original, am făcut câteva schimbări în implementarea noastră. În primul rând, valorile relevanţă (ρ) sunt obţinute din matricea termen-document (*A*) normalizată. În al doilea rând, folosim LSI pentru a calcula similaritatea (σ) între IE-uri, mai precis, matricea termen-termen [25]. (Autorii FCRN nu specifică modul în care se calculează similaritatea (σ).)

3.1.3 Fast Case Retrieval Net ++

În această secţiune este prezentată o variantă FCRN, propusă de noi, pe care o numim FCRN++.

Diferenţa dintre FCRN şi FCRN++ constă în modul în care sunt calculate funcţiile de relevanţă şi de similaritate. Pentru FCRN, relevanţa efectivă foloseşte matricea termen-termen ca funcţie de similaritate şi matricea termen-document normalizată, ca funcţie de relevanţă. Pentru FCRN++, funcţia de relevanţă este definită ca fiind similaritatea (cosinus) dintre documente şi termeni. Funcţia de similaritate este definită ca fiind similaritatea (cosinus) între termeni. Pentru calculul similarităţii cosinus, sunt utilizate coordonatele documentelor şi termenilor în spaţiu *k*-dimensional. Coordonatele documentelor în spaţiu *k*-dimensional sunt date de liniile matricei *Vk*, iar coordonatele termenilor sunt date de către liniile din matricea $Uk*Sk^{-1}$ [22]. Implementarea FCRN++, în Octave / Matlab, este prezentată în Figura 8.

Pas	Algoritmul FCRN++	Explicaţii
1	`% Vezi Figura 7, pasul 1.`	
2	`% Vezi Figura 7, pasul 2.`	
3	`Uk = U(:,1:k);` `Sk = S(1:k,1:k);` `Vk = V(:,1:k);` `Tk = Uk*inv(Sk);` `for j = 1 : size(Tk,1)` `  for i = 1 : size(Vk,1)` `  TD(j,i) =` `  sum(Tk(j,:) .*` `  Vk(i,:)) /` `  (norm(Tk(j,:))` `*  norm(Vk(i,:)));` `  end` `end` `for j = 1 : size(Tk,1)` `  for i = 1 : size(Tk,1)` `  TT(j,i) =` `  sum(Tk(j,:) .*` `  Tk(i,:)) /` `  (norm(Tk(j,:))` `*  norm(Tk(i,:)));` `  end` `end`	Matricea *Uk* conţine primele *k* coloane ale matricei *U*. *Sk* este o matrice pătratică de ordin *k* şi este formată din primele *k* linii şi coloane ale matricei *S*. *Vk* reţine coordonatele documentelor în spaţiu *k*-dimensional. *Tk* reţine coordonatele termenilor în spaţiu *k*-dimensional. În matricea *TD* este calculată similaritatea cosinus dintre termeni şi documente. În matricea *TT* este calculată similaritatea cosinus dintre termeni.
4	`E = TT * TD;` `% Vezi Figura 7, pasul 4.`	

Figura 8. Implementarea FCRN++

Ca input, FCRN++ utilizează aceleaşi surse, ca şi FCRN şi LSI. Acest algoritm generează acelaşi tip de output ca şi FCRN şi LSI.

4. Metodologia de evaluare

Această secțiune prezintă metodologia adoptată pentru obținerea rezultatelor experimentale din secțiunea următoare.

Am cerut mai multor persoane să evalueze pagini web, astfel încât să obținem un profil de utilizator pentru fiecare individ. După ce am obținut profilurile lor, am cerut totodată acestor persoane să: (1) caute pe web un subiect legat de profilul lor, (2) reordoneze rezultatele căutării în funcție de modul în care consideră de cuviință şi (3) să ne spună la ce rezultate se aşteaptă (şi în ce ordine).

Pentru algoritmul User Profile am variat parametrul *th*. Vă rugăm să rețineți că parametrul prag al algoritmilor LSI, FCRN şi FCRN++ a fost menținut fixat la o valoare de 0,4 (aleasă în urma unor experimente preliminare). De asemenea, pentru nucleul SVD, $k = 2$. Această valoare a fost şi ea aleasă experimental.

Pentru compararea rezultatelor am folosit mai multe, bine cunoscute, metrici de evaluare din domeniul Information Retrieval: Precision, R-Precision, Recall, şi F-measure [26]. Un Precision mare arată că au fost returnate rezultate mai relevante decât irelevante. Un Recall mare indică faptul că cele mai multe dintre rezultatele relevante au fost returnate. F-measure este o combinație dintre Precision şi Recall. Aceasta permite măsurarea preciziei. Rețineți că, deoarece lucrăm cu liste ordonate de pagini web, vom aplica metoda de calcul a Precision, Recall şi F-measure folosită pentru sistemele de Information Retrieval ordonate [27].

5. Experimente

Secțiunea următoare prezintă mai multe experimente în care se evaluează algoritmii de mai sus. În toate experimentele noastre, am folosit o valoare scăzută de prag pentru algoritmul User Profile pentru că nu am vrut să evaluăm capacitatea de categorisire a algoritmului implementat. Considerăm acest aspect dincolo de sfera de aplicare a acestei lucrări. De exemplu, un sistem de căutare web personalizată pentru clasificarea interogărilor utilizatorului este prezentat în [28]. În schimb, am permis utilizatorilor să creeze mai multe profiluri (prin urmare, o clasificare manuală). Mai mult decât atât, utilizatorii au fost instruiți să ofere doar feedback pozitiv. Pentru orice pagină web considerată irelevantă, nu s-a furnizat nicio evaluare. Am adoptat această abordare pentru că noi considerăm că este mai simplu pentru o persoană să evalueze doar ceea ce este interesant. Evident, acest lucru simplifică de asemenea procesul de rating.

Experimentul 1 a fost realizat cu pagini web scrise în limba română. Utilizatorul a fost interesat în găsirea unui răspuns la următoarea întrebare: "*Ce universități din România oferă programe de Master în matematică?*". Utilizatorul a inițializat profilul cu următoarele cuvinte: *matematici aplicate, finanțe, asigurări, analiză, matematică, modelar, matematică, mecanica mediilor continue*. Primele 23 de pagini web returnate de Google au fost luate în considerare. Vă rugăm să rețineți că pagina 24 a fost adăugată de către utilizator. A fost găsită ulterior de către acesta şi a fost considerată foarte reprezentativă. În prima etapă, toate cele 24 de pagini web au fost evaluate în raport cu profilul inițial al utilizatorului.

După etapa 1, o performanță similară a fost înregistrată între motorul de căutare şi algoritmul User Profile. LSI şi FCRN++ au avut aceeaşi performanță. Împreună cu FCRN, aceşti trei algoritmi au oferit performanțe semnificativ mai bune decât User Profile.
În etapa următoare, utilizatorul a dat rating-uri de 5 şi 4 stele la mai multe pagini. Ca urmare, profilul evoluat. Cele mai multe dintre cuvintele cheie de la profilul inițial s-au găsit în primele 30% dintre cuvintele cheie de la profilul evoluat. Cuvinte cheie noi au apărut, cu ponderi mai bune, pentru că au fost mai frecvent prezente în paginile web evaluate. De data aceasta, algoritmul User Profile a furnizat rezultate semnificativ mai bune, aşa cum se arată în Figura 9.

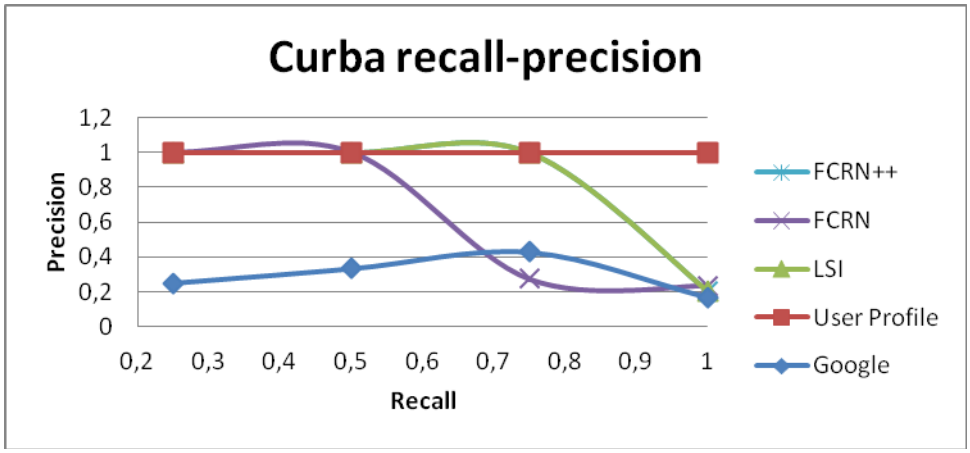

Figura 9. Curba recall-precision, la finalul experimentului 1

De fapt, User Profile a avut o curbă recall-precision ideală. LSI, FCRN şi FCRN++ au dat exact aceleaşi rezultate ca şi în pasul 1. Acest lucru poate fi explicat prin faptul că ei sunt algoritmi de recuperare. Mai mult decât atât, în ceea ce priveşte cuvintele cheie, profilul utilizatorului a rămas neschimbat de la o etapă la alta. Ponderile unor cuvinte cheie s-au schimbat deoarece utilizatorul a dat rating din nou unor pagini web. Totuşi, interogarea a rămas aceeaşi. User Profile a reuşit să beneficieze de feedback-ul utilizatorului, dar nu l-a reflectat suficient astfel încât modificări semnificative să fi fost prezente în matricea utilizată la calculul SVD.

Restul experimentelor folosesc limba engleză. În **experimentul 2** un utilizator a avut un interes în domeniul comerţului electronic. Fiind interesat să cumpere un smartphone cu tehnologie LTE, utilizatorul a iniţializat profilul cu pagina de web http://www.laptopmag.com/best-phones (şi i-a dat cel mai bun rating). După această etapă de iniţializare, o căutare Google a fost realizată folosind interogarea "*buy smartphones with LTE*". Utilizatorul a examinat (primele zece) pagini web returnate, a stabilit propria sa ordine (bazată pe interesele sale), precum a şi înregistrat, ordinea furnizată de algoritmul User Profile. Apoi, utilizatorul a mai efectuat un pas în care cel mai bun rating a fost dat paginilor cele mai reprezentative. Cu această intrare în plus, algoritmul User Profile a reuşit să dea rezultate foarte bune, aşa cum o arată Figura 10.

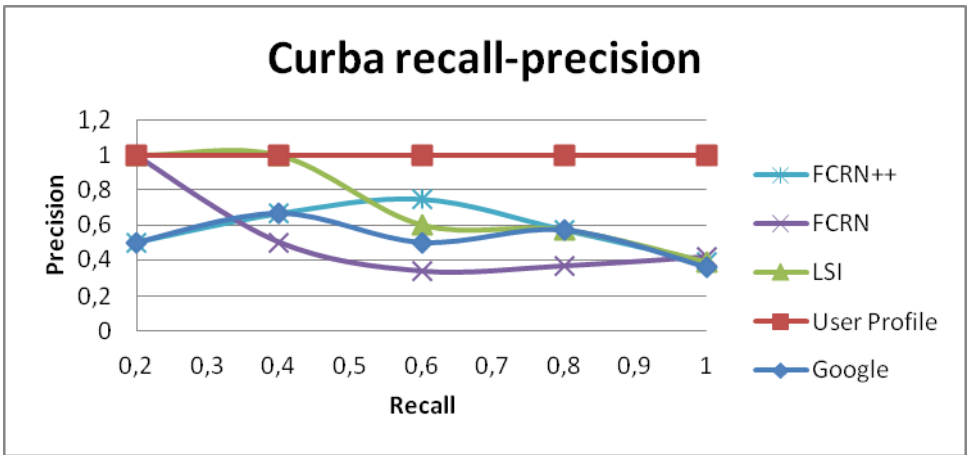

Figura 10. Curba recall-precision, la finalul experimentului 2

Dintre cei trei algoritmi de recuperare, LSI a dat cele mai bune rezultate în general. Comparând aceşti algoritmi cu profilul de utilizare, ajungem la aceeaşi concluzie ca cea menţionată pentru experimentul anterior.

În următorul **experiment**, un utilizator a efectuat următoarea căutare cu motorul de căutare Google: "*java latest version tutorials*". Primele 30 de rezultate furnizate de motorul de căutare au fost utilizate. Prin analiza paginilor web preluate, utilizatorul a început reiterarea lor şi popularea astfel a profilul de utilizator.

După configurarea inițială a profilului, experimentul a constat în cinci etape. Mai multe pagini au fost evaluate de către utilizator în prima etapă. Cele considerate cele mai relevante au primit patru şi cinci stele. Utilizatorul consideră pagina 1 ca fiind cea mai relevantă. Fiind nemulțumit că pagina 1 este prezentă doar a patra, utilizatorul i-a dat din nou rating de 5 (efectuând astfel etapa 2). Similar cu pasul 2, utilizatorul a efectuat următoarele trei etape prin oferirea de feedback bun şi neutru la alte trei pagini de web care au fost găsite reprezentative. După pasul 5, utilizatorul a considerat că a fost obținut rezultatul dorit. În timp ce în prima etapă Google a dat rezultate mai bune, pentru etapele 2-5, rezultatele sunt similare: algoritmul User Profile a depăşit clar motorul de căutare.

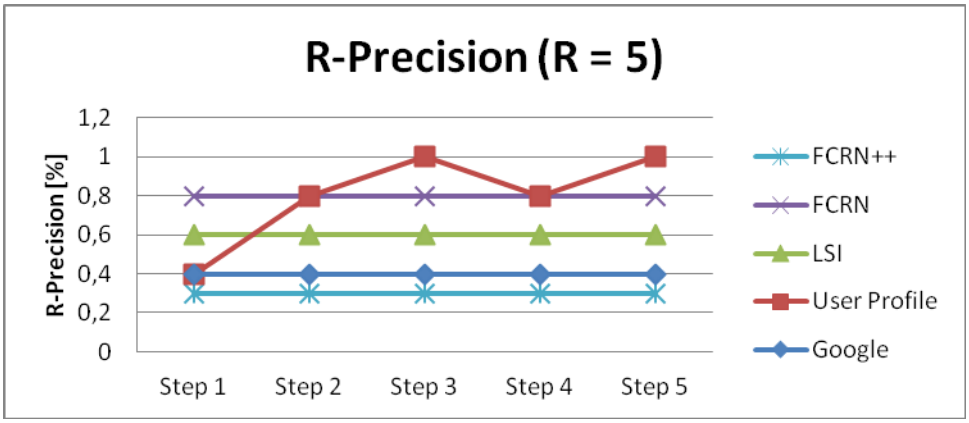

Figura 11. R-precision pentru 5 pagini web relevante (experiment 3)

Rezultatele de mai sus (Figura 11) arată că algoritmul User Profile reacționează bine la feedback-ul utilizatorului. Preferințele de utilizator sunt învățate şi prin urmare, un utilizator poate obține rezultate de căutare web mai adecvate pentru nevoile sale. Dintre cei trei algoritmi de recuperare, FCRN a dat cele mai bune rezultate. LSI şi FCRN++ au avut rezultate slabe. FCRN++ a fost mai rău chiar decât motorul de căutare. Credem că aceasta este, în principal, datorită faptului că multe dintre paginile web au avut mai puține informații textuale şi mai multe imagini.

La finalul acestui experiment, am dori să menționăm că profilul de utilizator obținut conține cuvinte cheie relevante, "java", fiind cel mai relevant.

În **experimentul** următor, vom încerca o analiză mai riguroasă. Pornim de la o interogare mai generală şi continuăm cu mai multe întrebări specifice. Această abordare permite şi motorului de căutare să răspundă mai bine aşteptărilor utilizatorului. Acest experiment constă într-o întrebare pentru care un utilizator doreşte să găsească un răspuns: *Which are the best software programming books about Java 8?* Utilizatorul a efectuat trei interogări succesive: (1) *best software programming books*, (2) *best java books* şi (3) *best java 8 books*. Rețineți că prima interogare este generală, următoarea este mai specifică şi a treia este şi mai specifică. După fiecare dintre cele trei interogări sunt considerate primele zece pagini web oferite de Google.

Experimentul a avut trei etape principale, fiecare corespunzând uneia dintre cele trei interogări. Fiecare set de pagini web a fost rearanjat cu profilul curent al utilizatorului. Apoi, acestea au fost evaluate, introduse în profil şi rearanjate. Scopul a fost de a măsura performanța algoritmului cu şi fără datele din noile pagini web. Cei trei algoritmi de recuperare au fost aplicați şi ei, folosind output-ul de la User Profile. Evident, la pasul 1, zece pagini web corespunzătoare primei interogări au fost introduse direct în profil.

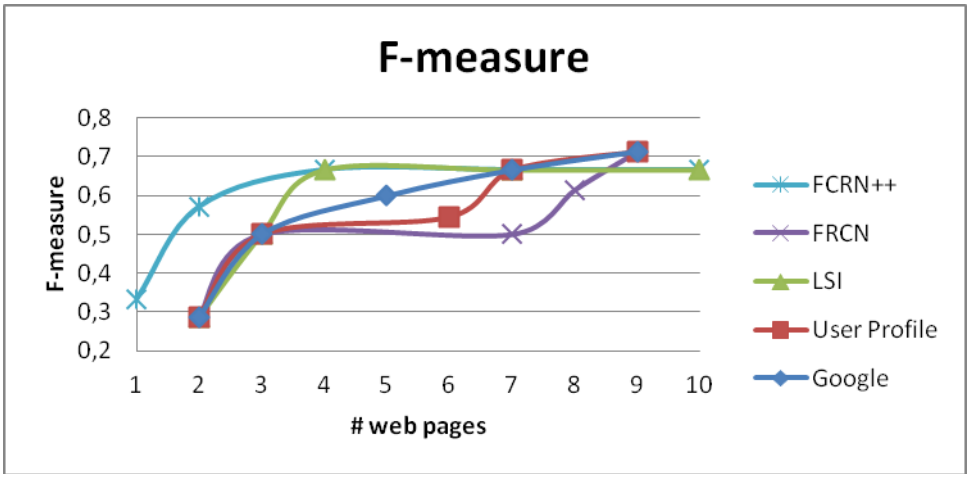

Figura 12. F-measure, la finalul experimentului 4

F-measure înregistrat în ultima etapă (Figura 12) a acestui experiment arată că algoritmi cum ar fi FCRN++ şi LSI, reuşesc să ofere rezultate mai bune decât restul algoritmilor, pentru primele cinci pagini de web.

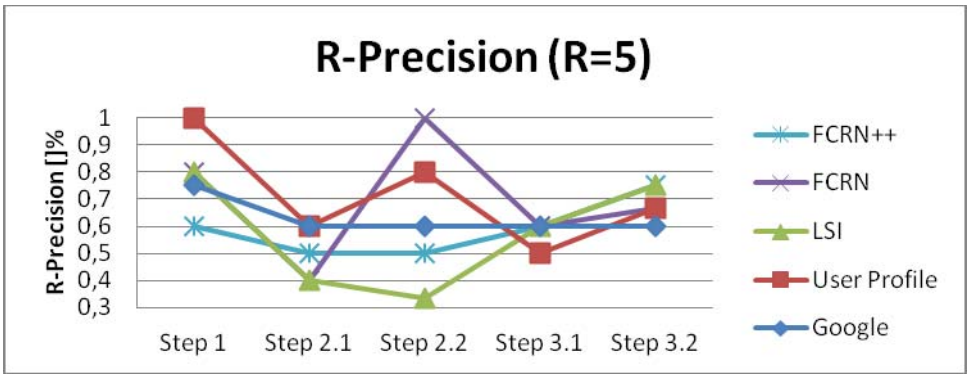

Figura 13. R-Precision pentru 5 pagini web relevante (experiment 4)

Evoluția generală a algoritmilor pentru întreg experimentul este prezentată în Figura 13. În aproape toate cazurile, toți algoritmii au dat rezultate mai bune după ce utilizatorul a oferit un feedback pentru paginile web care să fie reordonate. Motorul de căutare a avut aproape o evoluție constantă, chiar dacă utilizatorul a dat interogări tot mai specifice. La sfârşitul experimentului toți algoritmii au un R-Precision mai bun decât cel al motorului de căutare.

6. Concluzii şi cercetări viitoare

În această lucrare a fost prezentat un sistem de căutare personalizată pentru reordonarea rezultatelor de căutări web. Acest sistem este disponibil ca un plugin pentru browser-ul Mozilla Firefox şi funcționează cu motorul de căutare Google. Evident, poate fi extins pentru alte browsere şi motoare de căutare. Folosind tehnici cunoscute de NLP (cum ar fi stemming şi detectarea limbii), algoritmul User Profile este folosit pentru a afla şi stoca preferințele utilizatorilor. Experimentele noastre au arătat că profilul utilizatorului evoluează în direcția dorită. Mai mult decât atât, algoritmi cum ar fi LSI, FCRN şi FCRN++ (propus de noi) îmbunătățesc procesul de recuperare chiar din primele etape, în cazul în care profilul de utilizator nu este foarte consistent. Feedback-ul utilizatorului este esențial pentru buna performanță a algoritmilor.

Ca şi activitate viitoare, ne propunem să luăm în considerare alți algoritmi de profil ai utilizatorului, care se bazează pe feedback-ul implicit şi colaborativ al utilizatorilor. De asemenea, intenționăm să abordăm problema de expansiune a interogării. Clasificarea automată a datelor profilul utilizatorului este un alt aspect interesant.

BIBLIOGRAFIE

1. **PITKOW, J.; SCHÜTZE, H.; CASS, T.; COOLEY, R.; TURNBULL, D.; EDMONDS, A.; ADAR, E.; BREUEL, T.:** Personalized Search. *Commun ACM*, vol. 45, no. 9, Sep. 2002, pp. 50–55.
2. **SALTON, G.; MCGILL, M. J.:** Introduction to Modern Information Retrieval. New York, NY, USA: McGraw-Hill, Inc., 1986.
3. *** "Personalized Search for everyone," *Official Google Blog*.
4. *** "Making search yours | Search Blog." [Online]. Available: http://blogs.bing.com/search/2011/02/10/making-search-yours/. [Accessed: 23-Oct-2014].
5. **WRITER, M. H. S.; NEWS, C.:** Yahoo debuts personalized search - CNET News. *CNET*. [Online]. Available: http://news.cnet.com/Yahoo-debuts-personalized-search/2100-1038_3-5686585.html. [Accessed: 23-Oct-2014].
6. **HANNAK, A.; SAPIEZYNSKI, P.; MOLAVI KAKHKI, A.; KRISHNAMURTHY, B.; LAZER, D.; MISLOVE, A.; WILSON, C.:** Measuring Personalization of Web Search. In Proceedings of the 22Nd International Conference on World Wide Web, Republic and Canton of Geneva, Switzerland, 2013, pp. 527–538.
7. **WIDYANTORO**, **D. H.:** Dynamic Modeling and Learning User Profile in Personalized News Agent. Texas A&M University, 1999.
8. **SUGIYAMA, K.; HATANO, K.; YOSHIKAWA, M.:** Adaptive Web Search Based on User Profile Constructed Without Any Effort from Users. In Proceedings of the 13th International Conference on World Wide Web, New York, NY, USA, 2004, pp. 675–684.
9. **TEEVAN, J.; DUMAIS, S. T.; HORVITZ, E.:** Personalizing Search via Automated Analysis of Interests and Activities. In Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 2005, pp. 449–456.
10. **GAUCH, S.; CHAFFEE, J.; PRETSCHNER, A.:** Ontology-based Personalized Search and Browsing. Web Intelli Agent Sys, vol. 1, no. 3–4, Dec. 2003, pp. 219–234.
11. **QIU, F.; CHO, J.:** Automatic Identification of User Interest for Personalized Search. In Proceedings of the 15th International Conference on World Wide Web, New York, NY, USA, 2006, pp. 727–736.
12. **SUN, J.-T.; ZENG, H.-J.; LIU, H.; LU, Y.; CHEN, Z.:** CubeSVD: A Novel Approach to Personalized Web Search. In Proceedings of the 14th International Conference on World Wide Web, New York, NY, USA, 2005, pp. 382–390.
13. **SHEN, X.; TAN, B.; ZHAI, C.:** Implicit User Modeling for Personalized Search. In Proceedings of the 14th ACM International Conference on Information and Knowledge Management, New York, NY, USA, 2005, pp. 824–831.
14. **TAN, B.; SHEN, X.; ZHAI, C.:** Mining Long-term Search History to Improve Search Accuracy. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2006, pp. 718–723.
15. **PARISER, E.:** The Filter Bubble: How the New Personalized Web Is Changing What We Read and How We Think. Penguin, 2011.
16. **KOEHN**, **P.:** Europarl: A Parallel Corpus for Statistical Machine Translation.
17. *** Coursera - Language Modeling, *Coursera*. [Online]. Available: https://class.coursera.org/nlp/lecture/17. [Accessed: 16-Oct-2014].
18. **PORTER, M. F.:** Readings in Information Retrieval. K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 313–316.

19. **FIELDING, R. T.; TAYLOR**, **R. N.:** Principled Design of the Modern Web Architecture. ACM Trans Internet Technol, vol. 2, no. 2, May 2002, pp. 115–150.

20. **BERRY, M.; DUMAIS, S.; O'BRIEN, G.:** Using Linear Algebra for Intelligent Information Retrieval. SIAM Rev., vol. 37, no. 4, Dec. 1995, pp. 573–595.

21. **SALTON, G.: WONG, A.; YANG**, **C. S.:** A Vector Space Model for Automatic Indexing. Commun ACM, vol. 18, no. 11, Nov. 1975, pp. 613–620.

22. **BERRY, M. W.; DRMAČ, Z.; JESSUP**, **E. AND R.:** Matrices, vector spaces, and information retrieval. SIAM Rev., vol. 41, 1999, pp. 335–362.

23. **RICCI, F.; ROKACH, L.; SHAPIRA, B.; KANTOR**, **P. B. Eds.:** Recommender Systems Handbook, 2011 edition. New York: Springer, 2010.

24. **LENZ, M.; BURKHARD, H.-D.:** Case retrieval nets: Basic ideas and extensions. In KI-96*:* Advances in Artificial Intelligence, G. Görz and S. Hölldobler, Eds. Springer Berlin Heidelberg, 1996, pp. 227–239.

25. **KONTOSTATHIS, A.; POTTENGER, W. M.:** A Mathematical View of Latent Semantic Indexing: Tracing Term Co-Occurrences, 2002.

26. **RIJSBERGEN, C. J. V.:** Information Retrieval. 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

27. **TEUFEL, S.:** An Overview of Evaluation Methods in TREC Ad Hoc Information Retrieval and TREC Question Answering. In Evaluation of Text and Speech Systems, P. L. Dybkjær, H. Hemsen, and P. W. Minker, Eds. Springer Netherlands, 2007, pp. 163–186.

28. **LIU, F.:** Personalized web search by mapping user queries to categories. 2002, pp. 558–565.