# Pràctica 3: User Relevance Feedback

### Implementació de Rocchio’s rule

Per a l’implementació del fitxer Rochio.py hem usat el codi de SearchIndexWeigth.py fusionat també amb parts del TFIDF.py de la sessió anterior. D’aquest últim codi hem fet servir les funcions *toTFIDF*, *normalize*, i per últim també *document\_term\_vector*.

En aquesta sessió hem treballat amb diccionaris en comptes de vectors, això ha suposat més treball ja que hem hagut de canviar les implementacions originals del toTFIDF i normalize importades del fitxer de la sessió anterior. Malgrat aquesta feina extra que ha suposat, hem aconseguit una millora en eficiència, ja que fusionar vectors ordenats suposa un cost computacional més gran. Els cost del merging de vectors és O(n1 + n2), en canvi en el cas de fer servir diccionaris només l’hem de recórrer i per tant el cost computacional és O(min(n1, n2)).

Apart del codi importat d’aquests fitxers també hem creat noves funcions per a implementar la fórmula de Rocchio i per treballar amb diccionaris, aquestes són:

* *rocchio*: encarregada de implementar la funció de Rocchio donada una query (passada com a paràmetre de la funció), una alpha, una beta, nrounds i R (variables global que anem modificant per als diferents experiments) i el nombre de documents rellevants (representat per una k a la fórmula però corresponent a nhits, valor que entrem com a flag per terminal)



El que estem fent bàsicament és afegir a la query inicial nous termes i ponderant la query antiga amb el paràmetre alpha i els nhits termes més rellevants del tfidf amb el paràmetre beta

* *query\_a\_diccionari* i *diccionary\_a\_query*: tal i com indica el nom de les funcions aquestes s’encarreguen de transformar les queries a diccionaris i viceversa.

### Experiments i conclusions

Un cop implementat el codi per observar el User Relevance Feedback utilitzant la regla de Rocchio, hem realitzat diversos experiments que ens ajudin a entendre millor el seu funcionament i la influència dels paràmetres de la funció sobre els resultats obtinguts.

Per fer els experiments hem utilitzat l’índex *news,* obtingut a partir de la carpeta *20\_news\_groups,* ja que hem considerat rellevant la seva diversitat de temàtiques en els diferents documents que conté.

Sempre fem servir els mateixos valors per als paràmetres de la  *Rocchio’s rule:*

* alpha = 1
* beta = 1
* R = 4
* nrounds = 5
* nhits = 10

Excepte alguns casos de l’experiment 4 que s’explica explícitament quins canvis hem fet.

* 1. Experiment inicial

En aquest primer experiment hem volgut testejar les queries que se’ns proporciona a l’enunciat com a exemple, amb uns paràmetres de alpha i beta igualats, per no influir en la importància dels termes. Els resultats obtinguts són els següents:

toronto nyc → 1 document amb SCORE=32.3232

toronto nyc^2 → 1 document amb SCORE=36.62403

toronto^2 nyc → 1 document amb SCORE=34.477364

Com podem veure totes tres queries ens retornen 1 sol document rellevant. Les puntuacions d’aquest varien segons la importància que se li atribueix als termes inicials, però la quantitat de documents no canvia ja que suposem que és l’únic en el conjunt que conté ambdues paraules a la vegada.

* 1. Experiment sobre l’importància inicial

Per aquest segon experiment, hem buscat una query que no tingués problema en cercar documents que continguessin les paraules en qüestió. Hem utilitzat els termes “a” i “the”, que són altament freqüents en la llengua anglesa. En concret, hem volgut experimentar sobre l’importància inicial que se li dona als mots en la comanda per terminal, per veure com varia la quantitat de documents retornats i les puntuacions. Els primers resultats obtinguts han estat:

a the → 18 documents

R termes en la query en la primera ronda:

['a^0.7300871989062048', 'the^0.7196513968778471', 'barrel^0.1231102803783805', 'interprets^0.10130444836451882']

R termes en la query en la última ronda:

['barrel^0.5142940958879993', 'interprets^0.4693811961118566', 'a^0.3667705512203818', 'the^0.35542612777986005']

Com podem veure en la primera ronda d’aplicació de la regla de Rocchio hem obtingut una query on els termes més rellevants han estat els de la query inicial, afegint dos nous termes “barrel” i “interprets” que a priori no tenen molta relació. No ens sorprèn ja que els mots utilitzats al principi són molt generals i careixen de significat propi com per estar lligades a una certa temàtica i que es puguin trobar paraules relacionades. A la última ronda veiem com aquestes paraules inicials han perdut rellevància dins la query, novament per ser paraules molt freqüents en tot el conjunt de documents a l’índex.

A continuació, hem volgut experimentar amb importàncies diferents inicialment i observar com evoluciona la nova query de termes rellevants:

a^2 the →

R termes en la query en la última ronda:

['a^0.5035983563336958', 'barrel^0.500203506468623', 'interprets^0.45707101945144496', 'the^0.2618963035023346']

a^6 the →

R termes en la query en la última ronda:

['a^0.7143359100111929', 'barrel^0.47271910996636274', 'interprets^0.43293920505138433', 'thug^0.1763751071379798']

Com podem veure, quanta més importància inicial li donem a un terme, més importància té en les queries generades a partir de la regla de Rocchio. Posant una importància de 2 a la “a”, observem que a la última ronda segueix sent la paraula amb més pes. L’experiment amb importància 6 a la “a” ens mostra el mateix, amb la diferència que el terme “the” l’hem acabat perdent i s’ha substituït per un nou terme, possiblement perquè la seva rellevància ha decaigut i ha estat superada.

* 1. Experiment sobre la combinació de paraules

En aquest tercer experiment hem volgut veure l’efecte de fer queries independents de dues paraules que no tenen res a veure per separat i després veure quin resultat dona la combinació d’ambdues juntes i després fer-ho per a dues que si són complementàries. Hem usat els termes “ball” i “science” per a aquest experiment ja que hem pensat que el primer mostraria documents i afegiria termes a la query relacionats amb l’esport i el segon amb la ciència.

ball -> 6 documents

['ball^0.8993253960509465', 'infield^0.40206570729497915', 'fly^0.38984395723798565', 'runner^0.24527538599066165']

science -> 14 documents

['science^1.2387460055919048', 'origins^0.6704811005674061', 'percent^0.523927336049692', 'rawlins^0.3644312921653081']

ball science -> 0 documents

['ball^0.49377418243001636', 'science^0.47760127405376523', 'demers^0.055941237077839705', 'addrs^0.03969481517350357']

Com nosaltres suposàvem en un començament les queries individuals sí que han suggerit documents i han afegit termes lògics a les queries però com predeiem la combinació d’ambdues no ha pogut recomanar cap document i els termes afegits a la query són “randoms” no tenen a veure amb els termes originals.

Ara volem comprovar que en cas contrari, quan els termes de la query tenen relació entre sí, aquesta sí que suggereix documents i els termes que s’afegeixen a la query tenen a veure amb els originals. Per a aquesta part usarem els termes “astro” i “science” amb les quals esperavem resultats relacionats amb l’astronomia/espai.

science astro

['astro^0.4597414339706039', 'science^0.4426566839891184', 'space^0.27236139659500813', 'nasa^0.18876832060723692']

6 documents

documents de la carpeta: 20\_newsgroup/sci.space

Com es pot veure en els resultats els documents i els nous termes de la query són tots relacionats amb la ciència i astronomia. Així que es confirmen les nostres hiòtesis.

* 1. Experiment modificant els paràmetres de Rocchio’s rule

Per aquest últim experiment hem jugat amb els paràmetres que usa la funció Rocchio. Hem utilitzat una query inicial d’una única paraula “windows” que ens retornava forces documents, i d’aquesta manera hem vist com variava aquesta quantitat quan li canviaven els paràmetres *nhits, nrounds, alpha, beta, R.*

Variació nhits

**nhits** = 10 → 29 Documents

**nhits** = 20 → 44 Documents

**nhits** = 5 → 10 Documents

Observem que, quant més gran és el nombre *nhits,* més documents ens retorna la consulta. Això és degut a que aquest paràmetre determina el nombre de documents que es consideren per calcular el vector de tfidf dels seus termes, en la fórmula de Rocchio(k). Per tant, si augmentem *nhits* estem incrementant el nombre de documents que es retornen, pujant així el recall, ja que abarquem més fitxers que contenen les paraules en les que estem interessats, però en algun punt disminuint la precisió (agafem documents que en realitat no són rellevants).

Variació nrounds

**nrounds** = 5 → SCORE màxima = 53.124702

**nrounds** = 10 → SCORE màxima = 123.20552

Modificant aquest paràmetre podem observar que es veu afectada la score dels documents que ens retorna el resultat. En ambdues línies de comanda la sortida és dels mateixos 10 fitxers, amb la query de paraules igual (variant els pesos). Concluïm que com aquest paràmetre controla les vegades que s’aplica la regla de Rocchio, el que fa es augmentar la puntuació cada vegada més, ja que els pesos es van sumant al llarg de les aplicacions.

Variació alpha i beta

**alpha** = 1 →

['windows^1.102843585226092', 'microsoft^0.44164171635351324', 'dos^0.34494727773288886', '3.1^0.2986097972135835']

29 documents

**alpha** = 5 →

['windows^19.604715110768606', 'dos^0.9573089183387185', 'microsoft^0.9211787003434904', '3.1^0.6884689580346081']

29 documents

Observant els resultats es veu clarament que la ponderació dels termes de la query inicial (en aquest cas “windows”) prenen molta importància.

**beta** = 1 →

['windows^1.102843585226092', 'microsoft^0.44164171635351324', 'dos^0.34494727773288886', '3.1^0.2986097972135835']

29 documents

**beta** = 5 →

['strobl^2.1570634588644717', 'gmd.de^1.4197062820703907', 'windows^1.0168876996508458', 'microsoft^0.604632441634722']

4 documents

Com veiem en aquest cas, passa justament el contrari. Ara es dóna més ponderació als nous termes afegits a la query. Suposem que com que aquests aquests precisament no tenen gaire sentit (no són paraules coherents ni que es puguin relacionar amb windows) baixa el nombre de documents de la resolució.

Variació R

**R** = 4 → 10 documents

**R** = 8 → 1 documents

**R** = 2 → 356 document

Per últim si variem aquest paràmetre podem veure que quan major és el valor de R, menys documents ens retorna el resultat. Per una banda, tenim que si augmentem la R a 8, només obtenim un fitxer rellevant, ja que s’afegeixen més termes a la nova query i pocs documents contindran aquests mots. En canvi, si disminuim la R a 2 la sortida es dispara i tenim un resultat de 365 documents, ja que les paraules rellevants de la query són tan sols 2. En conclusió, podem dir que si augmentem la R disminuim el recall (però aconseguim documents amb més precisió), mentre que si l’augmentem abarquem més nombre de documents però podem trobar “falsos negatius”.