# SREC-I: prototipo de Sistema de RECuperación Inteligente

Juan Antonio MARTÍNEZ COMECHE

Escuela Universitaria de Biblioteconomía y Documentación
Universidad Complutense de Madrid
comeche@caelo.eubd.ucm.es

**RESUMEN**
Descripción de un prototipo de sistema de recuperación inteligente llamado SREC-I desarrollado en PROLOG. Se explican sus características generales, motivadas por su finalidad inicialmente didáctica. A continuación se detallan los principales módulos que lo componen y el código en PROLOG de dos de ellos.

**Palabras clave:** Sistema de recuperación de información; recuperación de información; código fuente; PROLOG.

## SREC- I: Prototype of Intelligent Information Retrieval System

**ABSTRACT**
A prototype of an intelligent information retrieval system called SREC-I and written in PROLOG is described. Its general characteristics, motivated by its educational finality, are explained. After that its principal modules are detailed and the source code of two of them in PROLOG is presented.

**Key Words:** Information Retrieval System; Information retrieval; source code; PROLOG

La recuperación de información, como área de estudio, tiene tras de sí más de cuarenta años de andadura. Desde sus comienzos el objetivo primordial no ha variado, de manera que satisfacer las necesidades informativas de los usuarios mostrando normalmente los documentos donde se hallará la información buscada y automatizar el proceso con la máxima eficacia y eficiencia, sigue siendo el reto fundamental de los numerosos investigadores que trabajan en este campo.

Durante estas décadas muchos enfoques distintos han sido ensayados, desde los ya tradicionales modelos booleano, vectorial o probabilístico, hasta los que podemos englobar bajo la denominación común de técnicas de inteligencia artificial, donde incluiremos las redes neuronales o los algoritmos genéticos.

Ya desde la primera conferencia TREC (Text REtrieval Conference), que tuvo lugar en Gaithersburg[1], Maryland, entre el 4 y el 6 de noviembre de 1992, se pres-

[1] HARMAN, D.K. Overview of the First Text REtrieval Conference (TREC-1). En HARMAN, D.K. (Ed.) The First Text Retrieval Conference (TREC-1). Gaithersburg, MD: NIST, 1993. (Special Publication 500-207).

tó una especial atención a la evaluación de los sistemas, y en concreto a la necesidad de contar con sistemas donde comprobar los resultados obtenidos y con unas colecciones de prueba en las que comparar las mejoras obtenidas con las técnicas sometidas a examen. Entre los diversos sistemas de recuperación de información (SRI) de acceso libre que se emplearán destacan SMART[2] y ZPRISE[3].

Pero el haber sido diseñados específicamente para la investigación y evaluación disminuye, sin embargo, su utilidad desde el punto de vista docente. Las colecciones de prueba que sirven de entrada a estos SRI, por ejemplo, reúnen todos los documentos en un único fichero de texto, reservando caracteres especiales para indicar el comienzo y el final de cada uno de ellos. Por el contrario, en los SRI reales la colección no es fija; lo habitual es que las altas y bajas de documentos sean constantes. A fin de que los alumnos también pudieran observar estos procesos de incorporación y eliminación en los fondos, SREC-I se diseñó de manera que cada documento se almacenase en un fichero independiente.

Por señalar otro aspecto importante, estos sistemas suelen exigir que las decisiones sobre parámetros de funcionamiento (método de cálculo de los pesos de los términos, por ejemplo) se adopten previamente a la ejecución del programa, de manera que mientras se ejecutan los usuarios/alumnos ni pueden intervenir ni se les comunica las técnicas concretas empleadas o los valores de los parámetros utilizados por los sistemas en ese momento. Gracias a ello los tiempos de computación pueden ser comparados y evaluados, aunque la utilidad didáctica quede seriamente mermada. Como inicialmente el enfoque prioritario en nuestro caso era el docente, SREC-I se diseñó de modo interactivo: el sistema advierte al alumno de muchos de los posibles errores cometidos por él al ejecutarlo, le explica su naturaleza y le ayuda a solucionarlos sin que se vea obligado a reiniciar el sistema. Además, le consulta expresamente sobre cada una de las técnicas y parámetros que puede adoptar en cada momento, con lo que se fuerza al alumno a ser mucho más consciente del modo de funcionamiento interno de los SRI.

A estas dos características señaladas debemos añadir una tercera no menos importante: SREC-I no se diseñó adoptando un modelo o enfoque específico a la hora de concebir la recuperación de información. Al contrario, la carga de documentos se realiza de modo que cualquiera de los tres modelos de recuperación denominados clásicos (booleano, vectorial y probabilístico) pueda en principio efectuarse con la colección, aunque todavía no se haya implementado el modelo probabilístico en SREC-I. Se decidió así en su momento pensando en que el alumno pudiera de esta manera adquirir conciencia de sus posibilidades si en el futuro debiese adoptar decisiones sobre los sistemas y las colecciones a su cargo.

Más aún, desde un principio se pensó en ir añadiendo al sistema nuevos módulos que aumentasen sus opciones de funcionamiento, principalmente provenientes

---

[2] El programa SMART (versión 11.0 comprimida) puede obtenerse en ftp://ftp.cs.cornell.edu/pub/smart [consultado el 12/12/2004]

[3] Para obtener el programa ZPRISE (versión 2.0) se requiere una contraseña otorgada por Darrin Dimmick (del NIST) en el correo electrónico darrin.dimmick@nist.gov.

del campo de la Inteligencia Artificial y más específicamente del área del Procesamiento del Lenguaje Natural. Es por ello que se le impuso el nombre de inteligente (SREC-I) y el motivo por el que se ha desarrollado íntegramente en PROLOG, considerando las capacidades específicas de este lenguaje para el desarrollo de programas relacionados con la manipulación del lenguaje natural y en general con la Inteligencia Artificial.

La primera de tales ampliaciones se halla en fase avanzada de desarrollo. Se trata de un lematizador para el castellano que espero vea la luz durante el año próximo. También se ha programado en PROLOG y existe la posibilidad de que SREC-I pueda emplearse para su evaluación. Su utilización o no con este propósito dependerá principalmente de la implementación de un algoritmo de carga que admita las colecciones de prueba existentes en un único fichero.

SREC-I ya ha sido probado durante los cursos académicos pasados con grupos de tercer curso de la Diplomatura en Biblioteconomía y Documentación (en la asignatura optativa Sistemas Avanzados de Tratamiento y Recuperación de la Información) y de segundo curso de la Licenciatura en Documentación (en la asignatura troncal Técnicas Avanzadas de Recuperación de Información), habiendo obtenido buena acogida y unos resultados en general satisfactorios. Esta circunstancia me anima a continuar con el proyecto e introducir mejoras en las deficiencias observadas.

Mi intención es ponerlo a disposición de todos los profesores que lo deseen a través de un portal web cuya puesta en marcha está prevista para el próximo curso. Mientras no sea posible efectuar la descarga del programa en su integridad, al menos podrá el lector interesado consultar en las páginas que siguen las características más destacadas de los módulos actuales de SREC-I y el código en PROLOG de dos de ellos, los que a mi juicio presentan mayor dificultad de desarrollo a pesar del ya numeroso material bibliográfico existente sobre PROLOG, pues aunque en ocasiones sea muy valioso, hasta donde conozco ninguno aborda específicamente la creación de código correspondiente a un SRI en dicho lenguaje.

## MÓDULO DE CARGA

- Objetivo: reducción de un documento en lenguaje natural –en inglés o en español– a sus palabras constituyentes sin acentuación, en minúsculas y sin palabras vacías, e incorporación optativa a una colección en un fichero independiente.

- Lista de palabras vacías por defecto:

  a,about,acertadamente,ademas,again,against,ago,ahora,al,algo,algun,alguna,algunas,alguno,algunos,all,alla,alli,alrededor,also,alta,altas,alto,altos,among,an,and,another,anothers,ante,antes,any,anymore,aquel,aquella,aquellas,aquello,aquellos,aqui,are,aren,around,as,asi,asimismo,at,aun,aunque,away,b,back,baja,bajas,bajo,bajos,be,because,before,begin,beginning,begins,behavior,behind,belong,below,best,between,bien,big,bigger,biggest,buen,buena,buenas,bueno,buenos,but,by,c,cabe,cabo,cada,can,cause,cese,c

ierta,ciertamente,ciertas,cierto,ciertos,como,con,consiguiente,contra,corta,cortas,corte,cortes,corto,cortos,cosa,cosas,cost,costo,costos,costs,course,cree,creo,cual,cuales,cualquier,cualquiera,cuando,cuanta,cuantas,cuanto,cuantos,cuya,cuyas,cuyo,cuyos,d,da,dada,dado,dan,dar,de,debe,deben,deber,deberia,deberian,del,desde,despues,di,diciendo,don,donde,dont,down,dr,durante,during,e,each,el,ella,ellas,ello,ellos,else,elsewere,embargo,en,end,enfrente,entonces,entrante,entre,era,eran,es,esa,esas,ese,eso,esos,esta,establece,establecen,estan,estando,estar,estas,este,esto,estos,estoy,et,&,etc,even,ever,every,everybody,f,fast,fe,felt,few,fija,fijas,fijo,fijos,fill,fin,final,finales,font,for,form,forma,formacion,formar,formas,formation,forms,frente,from,fue,fuera,fueran,full,g,go,good,gral,gran,grande,grandes,great,greather,h,ha,habia,habian,hace,hacen,hacer,hacia,had,half,han,has,hasta,have,hay,haya,hayan,he,hecho,hechos,her,here,high,higher,highest,him,himself,his,how,hui,i,if,in,input,into,ir,is,isn,it,its,itself,j,k,l,la,las,late,le,leida,les,less,lo,los,m,main,many,mas,me,meanwhile,menos,mi,mia,mias,mine,minus,mio,mios,mis,misma,mismas,mismo,mismos,more,most,mucha,muchas,mucho,muchos,muy,my,n,nada,ni,ningun,ninguna,ninguno,no,non,nor,nos,nosotras,nosotros,not,now,nuestra,nuestras,nuestro,nuestros,nul,o,of,off,on,once,os,other,others,otra,otras,otro,otros,out,outside,over,overall,own,owner,p,par,para,partir,pero,pesar,plus,poca,pocas,poco,pocos,podra,podran,podria,podrian,por,porque,puede,pueden,pues,q,que,quien,quienes,r,round,s,se,sea,sean,segun,sera,seran,she,si,sido,sien,siendo,sigue,sin,sino,siquiera,small,smallest,so,sobra,sobre,sola,solamente,solas,solo,solos,some,something,somethings,son,su,such,sur,sus,suya,suyas,suyo,suyos,t,tal,tambien,tan,tanta,tantas,tanto,tantos,than,that,the,their,them,then,they,thing,things,this,those,through,thus,to,toda,todas,todavia,todo,todos,too,total,tras,tu,tus,tuya,tuyas,tuyo,tuyos,u,un,una,unas,uno,unos,until,up,upon,v,va,van,ve,versus,vosotras,vosotros,vuestra,vuestras,vuestro,vuestros,w,was,we,were,what,when,where,whether,while,why,with,without,worth,worthwhile,would,x,y,ya,year,yes,yo,you,your,yours,z,0,1,2,3,4,5,6,7,8 y 9.

- Algoritmo optativo de reducción morfológica previsto inicialmente: Porter, tanto en su versión original para el inglés como traducido para el castellano. Este último se sustituirá por el lematizador para el castellano que se está desarrollando actualmente.

## MÓDULO DE CÁLCULO DEL VALOR DISCRIMINATORIO

- Objetivo: cálculo del valor discriminatorio aproximado de los términos de una colección empleando el método del centroide y hallando las similaridades mediante el coeficiente del Coseno.
- Característica destacada: Módulo no empleado habitualmente por el tiempo de computación que precisa: las pruebas efectuadas con sólo 685 términos en un Pentium IV a 2'8 GHz dieron un tiempo de ejecución de 25 minutos.
- El código en PROLOG es el siguiente:

```
valor_discriminatorio:-
    write('Introduzca ENTRE APOSTROFES –una sola comilla al principio y al final– la localizacion –desde el directorio raiz– y el nombre completo –incluyendo la extension– del archivo "valordiscrimtrmcol.txt", seguido de punto y pulse intro'),nl,
    read(ArchValordiscrimtrmcol),
```

```
(
exists_file(ArchValordiscrimtrmcol), !
;
write('No puede realizarse el calculo del valor discriminatorio de los terminos de la coleccion porque no se ha localizado el archivo "valordiscrimtrmcol" con extension "txt" donde anotar los resultados finales. Por favor, cree dicho archivo e inicie de nuevo el programa'),nl,
fail
),
write('Introduzca ENTRE APOSTROFES –una sola comilla al principio y al final– la localizacion –desde el directorio raiz– y el nombre completo –incluyendo la extension– del archivo "simcsncentroicol.txt", seguido de punto y pulse intro'),nl,
read(ArchSimcsncentroicol),
(
exists_file(ArchSimcsncentroicol), !
;
write('No puede realizarse el calculo del valor discriminatorio de los terminos de la coleccion porque no se ha localizado el archivo "simcsncentroicol" con extension "txt" donde anotar resultados parciales. Por favor, cree dicho archivo e inicie de nuevo el programa'),nl,
fail
),
write('Introduzca ENTRE APOSTROFES –una sola comilla al principio y al final– la localizacion –desde el directorio raiz– y el nombre completo –incluyendo la extension– del archivo "simcsncentroicolsintrm.txt", seguido de punto y pulse intro'),nl,
read(ArchSimcsncentroicolsintrm),
(
exists_file(ArchSimcsncentroicolsintrm), !
;
write('No puede realizarse el calculo del valor discriminatorio de los terminos de la coleccion porque no se ha localizado el archivo "simcsncentroicolsintrm" con extension "txt" donde anotar resultados parciales. Por favor, cree dicho archivo e inicie de nuevo el programa'),nl,
fail
),
write('Introduzca ENTRE APOSTROFES –una sola comilla al principio y al final– la localizacion –desde el directorio raiz– y el nombre completo –incluyendo la extension– del archivo "vcttfidfcol.txt", seguido de punto y pulse intro'),nl,
read(ArchVcttfidfcol),
(
exists_file(ArchVcttfidfcol), !
;
write('No puede realizarse el calculo del valor discriminatorio de los terminos de la coleccion porque no se ha localizado el archivo "vcttfidfcol" con extension "txt" de donde se toman los datos necesarios para la correcta ejecucion de este programa. Por favor, cree dicho archivo, ejecute el programa de vectorizacion de la coleccion y luego inicie de nuevo este programa'),nl,
fail
),
```

```
        open(ArchVcttfidfcol,read,E),
        lect_y_bd_entrada(E),
        close(E), !,
        setof(N,L ^ dc(N,L),LsNumDocs),
        prc_obt_ls_numeros_term(LsNumDocs,LsNumTrm),
        length(LsNumDocs,NumDocs),
        calculo_centroide(LsNumTrm,LsNumDocs,NumDocs,LsTrmCentroide),
        Term =.. [dc,0,LsTrmCentroide],
        assertz(Term),
        open(ArchSimcsncentroicol,write,S),
        sb_prc_calculo_sim_coseno(S,0,LsNumDocs,LsNumTrm),
        close(S), !,
        open(ArchSimcsncentroicol,read,E1),
        calculo_sim_media(E1,NumDocs,SimMediaCentroiCol),
        close(E1), !,

calculo_valor_discriminatorio_col(LsNumTrm,LsNumTrm,LsNumDocs,NumDocs,Sim-
MediaCentroiCol,ArchValordiscrimtrmcol,ArchSimcsncentroicolsintrm),
        abolish(dc,2).


calculo_centroide(LsNumTrm,LsNumDocs,NumDocs,LsTrmCentroide):-
        LsTrmCentroideAux = [],

calculo_centroide(LsNumTrm,LsNumDocs,NumDocs,LsTrmCentroideAux,LsTrmCentroide).


calculo_centroide([],_,_,LsTrmCentroide,LsTrmCentroide):- !.

calculo_centroide([Cabeza|Cola],LsNumDocs,NumDocs,LsTrmCentroideAux,LsTrmCen-
troide):–
        pr_obt_sumatorio_un_trm(Cabeza,LsNumDocs,SumatorioTrm),
        ValorTrmCentroide is SumatorioTrm/NumDocs,
        ComponenteCentroide = Cabeza/ValorTrmCentroide,
        append(LsTrmCentroideAux,[ComponenteCentroide],LsTrmCentroideAct),
        calculo_centroide(Cola,LsNumDocs,NumDocs,LsTrmCentroideAct,LsTrmCentroide).


pr_obt_sumatorio_un_trm(NTrm,LsNumDocs,SumatorioTrm):-
        SumatorioTrmAux = 0,
        pr_obt_sumatorio_un_trm(NTrm,LsNumDocs,SumatorioTrmAux,SumatorioTrm).


pr_obt_sumatorio_un_trm(_,[],SumatorioTrm,SumatorioTrm):- !.

pr_obt_sumatorio_un_trm(NTrm,[Ndc|Cola],SumatorioTrmAux,SumatorioTrm):-
        dc(Ndc,LsTrm),
        (
```

```
        member(NTrm/ValorTrm,LsTrm), !,
        SumatorioTrmParc is SumatorioTrmAux+ValorTrm,
        pr_obt_sumatorio_un_trm(NTrm,Cola,SumatorioTrmParc,SumatorioTrm)
        ;
        pr_obt_sumatorio_un_trm(NTrm,Cola,SumatorioTrmAux,SumatorioTrm)
        ).


sb_prc_calculo_sim_coseno(_,_,[],_):- !.

sb_prc_calculo_sim_coseno(S,Cabeza,[X|Resto],LsNumTrm):-
        sb_prc_calculo_sim_coseno1(S,Cabeza,X,LsNumTrm),
        sb_prc_calculo_sim_coseno(S,Cabeza,Resto,LsNumTrm).



sb_prc_calculo_sim_coseno1(S,NdcA,NdcB,LsNumTrm):-
        dc(NdcA,LsA),
        dc(NdcB,LsB),
        calculo_prod_escalar(LsA,LsB,LsNumTrm,ProdEsc),
        NumeradorFormula = ProdEsc,
        sumatorio_cuadrados_trm_doc(LsA,SumatCuadraTrmDocA),
        sumatorio_cuadrados_trm_doc(LsB,SumatCuadraTrmDocB),
        ProdSumatorios is SumatCuadraTrmDocA*SumatCuadraTrmDocB,
        DenominadorFormula is sqrt(ProdSumatorios),
        ValorSimCoseno is NumeradorFormula/DenominadorFormula,
        Term =.. [simcos,NdcA,NdcB,ValorSimCoseno],
        write(S,Term),
        write(S,'.'),
        nl(S).



calculo_valor_discriminatorio_col([],_,_,_,_,_,_):- !.

calculo_valor_discriminatorio_col([NTrm|Resto],LsNumTrm,LsNumDocs,NumDocs,Sim-
MediaCentroiCol,ArchValordiscrimtrmcol,ArchSimcsncentroicolsintrm):-

calculo_valor_discriminatorio_un_trm(NTrm,LsNumTrm,LsNumDocs,NumDocs,SimMe-
diaCentroiCol,ArchValordiscrimtrmcol,ArchSimcsncentroicolsintrm),
        calculo_valor_discriminatorio_col(Resto,LsNumTrm,LsNumDocs,NumDocs,Sim-
MediaCentroiCol,ArchValordiscrimtrmcol,ArchSimcsncentroicolsintrm).



calculo_valor_discriminatorio_un_trm(NTrm,LsNumTrm,LsNumDocs,NumDocs,SimMe-
diaCentroiCol,ArchValordiscrimtrmcol,ArchSimcsncentroicolsintrm):-
        delete(LsNumTrm,NTrm,LsNumTrmMenosUno),
```

```
        open(ArchSimcsncentroicolsintrm,write,S2),
        sb_prc_calculo_sim_coseno_ref(S2,0,LsNumDocs,LsNumTrmMenosUno,NTrm),
        close(S2), !,
        open(ArchSimcsncentroicolsintrm,read,E2),
        calculo_sim_media(E2,NumDocs,SimMediaCentroiColSinTrm),
        close(E2), !,
        ValorDiscriminatorio is SimMediaCentroiColSinTrm-SimMediaCentroiCol,
        Term =.. [vdt,NTrm,ValorDiscriminatorio],
        nl,
        write('Calculado el valor discriminatorio del termino '),
        write(NTrm),nl,nl,
        write('Hallando el valor discriminatorio del siguiente termino...'),
        nl,nl,nl,
        open(ArchValordiscrimtrmcol,append,S3),
        write(S3,Term),
        write(S3,'.'),
        nl(S3),
        close(S3), !.



sb_prc_calculo_sim_coseno_ref(_,_,[],_,_):- !.

sb_prc_calculo_sim_coseno_ref(S,Cabeza,[X|Resto],LsNumTrm,NTrm):-
        sb_prc_calculo_sim_coseno1_ref(S,Cabeza,X,LsNumTrm,NTrm),
        sb_prc_calculo_sim_coseno_ref(S,Cabeza,Resto,LsNumTrm,NTrm).



sb_prc_calculo_sim_coseno1_ref(S,NdcA,NdcB,LsNumTrm,NTrm):-
        dc(NdcA,LsA),
        dc(NdcB,LsB),
        calculo_prod_escalar(LsA,LsB,LsNumTrm,ProdEsc),
        NumeradorFormula = ProdEsc,
        delete(LsA,NTrm/_,LsARef),
        delete(LsB,NTrm/_,LsBRef),
        sumatorio_cuadrados_trm_doc(LsARef,SumatCuadraTrmDocA),
        sumatorio_cuadrados_trm_doc(LsBRef,SumatCuadraTrmDocB),
        ProdSumatorios is SumatCuadraTrmDocA*SumatCuadraTrmDocB,
        DenominadorFormula is sqrt(ProdSumatorios),
        ValorSimCoseno is NumeradorFormula/DenominadorFormula,
        Term =.. [simcos,NdcA,NdcB,ValorSimCoseno],
        write(S,Term),
        write(S,'.'),
        nl(S).


calculo_prod_escalar(LsA,LsB,LsNumTrm,ProdEsc):-
```

```
        ProdEscAnt = 0,
        sb_pr_calculo_prod_escalar(LsA,LsB,LsNumTrm,ProdEscAnt,ProdEsc).



sb_pr_calculo_prod_escalar(_,_,[],ProdEsc,ProdEsc):- !.

sb_pr_calculo_prod_escalar(LsA,LsB,[NTrm|Cola],ProdEscAnt,ProdEsc):-
        member(NTrm/ValorTrmA,LsA),
        member(NTrm/ValorTrmB,LsB), !,
        ProdEscAct is ValorTrmA*ValorTrmB,
        ProdEscParc is ProdEscAnt+ProdEscAct,
        sb_pr_calculo_prod_escalar(LsA,LsB,Cola,ProdEscParc,ProdEsc)
        ;
        sb_pr_calculo_prod_escalar(LsA,LsB,Cola,ProdEscAnt,ProdEsc).


sumatorio_cuadrados_trm_doc(Ls,SumatCuadraTrmDoc):-
        SumatCuadraTrmAnt = 0,
        sumatorio_cuadrados_trm_doc(Ls,SumatCuadraTrmAnt,SumatCuadraTrmDoc).


sumatorio_cuadrados_trm_doc([],SumatCuadraTrmDoc,SumatCuadraTrmDoc):- !.

sumatorio_cuadrados_trm_doc([Cabeza|Cola],SumatCuadraTrmAnt,SumatCuadraTrm-
Doc):-
        Cabeza = _/ValorTrm,
        CuadradoValorTrm is ValorTrm*ValorTrm,
        SumatCuadraTrmParc is SumatCuadraTrmAnt+CuadradoValorTrm,
        sumatorio_cuadrados_trm_doc(Cola,SumatCuadraTrmParc,SumatCuadraTrmDoc).


lect_y_bd_entrada(E):-
        repeat,
        read(E,X),
        ( X == end_of_file, !
        ;
        assertz(X),
        fail).


prc_obt_ls_numeros_term(LsNumDocs,LsNumTrm):-
        LsNumTrmAux = [],
        prc_obt_ls_numeros_term(LsNumDocs,LsNumTrmAux,LsNumTrm).

prc_obt_ls_numeros_term([],LsNumTrm,LsNumTrm):- !.
prc_obt_ls_numeros_term([Cabeza|Cola],LsNumTrmAux,LsNumTrm):-
        dc(Cabeza,Ls),
```

```
        findall(NT,member(NT/_,Ls),LsNumTrmAct),
        append(LsNumTrmAux,LsNumTrmAct,LsNumTrmParc1),
        setof(X,member(X,LsNumTrmParc1),LsNumTrmParc2),
        prc_obt_ls_numeros_term(Cola,LsNumTrmParc2,LsNumTrm).



calculo_sim_media(E,NumDocs,SimMedia):-
        SumatSimAnt = 0,
        calculo_sim_media(E,NumDocs,SumatSimAnt,SimMedia).

calculo_sim_media(E,NumDocs,SumatSimAnt,SimMedia):-
        read(E,X),
        (
        X == end_of_file, !,
        SimMedia is SumatSimAnt/NumDocs
        ;
        X = simcos(_,_,ValorSim),
        SumatSimAct is SumatSimAnt+ValorSim,
        calculo_sim_media(E,NumDocs,SumatSimAct,SimMedia)
        ).
```

**Módulo de vectorización**

- Objetivo: ponderación de los términos de indización y representación de los documentos de la colección con tres opciones: binaria, tf.idf y señal-ruido.
- Característica destacada: una vez realizada una modalidad de vectorización es posible realizar otra posterior sin necesidad de reiniciar el sistema.

**Módulo de interrogación booleana**

- Objetivo: Realización de una consulta booleana del sistema y muestra de los resultados de la búsqueda.
- El código en Prolog es el siguiente:

```
recuperacion_booleana:-
        write('Introduzca la consulta, siguiendo las correspondientes normas de formulacion
booleana:'),nl,
        read(Cons),
        prc_interrog(Cons).


prc_interrog(Cons):-
        LsOperacPend = [],
        LsExpCompr = [],
        sb_prc_interrog(Cons,LsOperacPend,LsExpCompr).
```

```
sb_prc_interrog(Formula,LsOperacPend,LsExpCompr):-
        detec_elem_form(Formula,SbLsFormula),
        append([SbLsFormula],LsOperacPend,LsOperacPendAct),
        SbLsFormula = [_,prc(_,LsExp)],
        append(LsExp,LsExpCompr,LsExpComprAct),
        prc_compr_exp(LsOperacPendAct,LsExpComprAct).



calc_operac_pend(LsOperac):-
        lct_y_abd_todo_arch('C:/Mis documentos/prolog/pldtcol.txt'),
        pr_calc_operac_pend(LsOperac).


pr_calc_operac_pend(LsOperac):-
        length(LsOperac,1), !,
        LsOperac = [SbLsFormula],
        calc_operac(SbLsFormula),
        SbLsFormula = [Formula,_],
        formula(Formula,_,_,SetRsl),
        length(SetRsl,N),
        (
        N =:= 0, !,
        write('No existen documentos en el sistema que satisfagan su consulta'),
        nl,nl
        ;
        write('Existen '),write(N), write(' documentos en el sistema que satisfacen su consul-
ta:'),
        nl,
        escribe_lista(SetRsl)
        ),nl,
        abolish(pl,5),
        abolish(formula,4),
        write('Si desea realizar otra consulta, teclee c minuscula seguida de punto e intro'),nl,
        write('Si desea abandonar el programa, teclee s minuscula seguida de punto e intro'),
        nl,
        accion
        ;
        LsOperac = [Cabeza|Cola],
        calc_operac(Cabeza),
        pr_calc_operac_pend(Cola).



accion:-
        read(Resp),
        (
        Resp == c, !,
```

```
        recuperacion_booleana
        ;
        Resp == s, !
        ;
        write('Respuesta desconocida. Por favor, introduzcala de nuevo'),nl,
        accion
        ).

detec_elem_form(Formula,SbLsFormula):-
        name(Formula,LsCarForm),
        detec_error_y_oper_simples(Formula,LsCarForm,SbLsFormula).


detec_error_y_oper_simples(Formula,[Cabeza|Cola],SbLsFormula):-
        Cabeza \== 40, !,
        mensaje_error(Formula)
        ;
        last(X,Cola),
        X \== 41, !,
        mensaje_error(Formula)
        ;
        sb_prc_detec_oper_simples(Formula,Cola,SbLsFormula).


sb_prc_detec_oper_simples(Formula,[Cabeza|Resto],SbLsFormula):-
        Cabeza == 110,
        Resto = [111|Cola], !,
        append([32],LsParc,Cola),
        append(LsCarExa,[41],LsParc),
        name(Exa,LsCarExa),
        LsExp = [Exa],
        SbLsFormula = [Formula,prc(no,LsExp)]
        ;
        Cabeza \== 40, !,
        append(LsCarExa,[32|Cola],[Cabeza|Resto]),
        name(Exa,LsCarExa),
        append(LsCarOperador,[32|LsParc],Cola),
        name(Operador,LsCarOperador),
        append(LsCarExb,[41],LsParc),
        name(Exb,LsCarExb),
        LsExp = [Exa,Exb],
        SbLsFormula = [Formula,prc(Operador,LsExp)]
        ;
        sb_prc_detec_form_compleja(Formula,[Cabeza|Resto],SbLsFormula), !.


sb_prc_detec_form_compleja(Formula,[_|Fin],SbLsFormula):-
        LsCarExa = [40],
```

```
        Num = 1,
        detec_formula_compleja(Formula,Fin,Num,LsCarExa,SbLsFormula).



detec_formula_compleja(Formula,[Cabeza|Resto],Num,LsCarExa,SbLsFormula):-
        Num =:= 0, !,
        name(Exa,LsCarExa),
        append(LsCarOperador,[32|Cola],Resto),
        name(Operador,LsCarOperador),
        append(LsCarExb,[41],Cola),
        name(Exb,LsCarExb),
        LsExp = [Exa,Exb],
        SbLsFormula = [Formula,prc(Operador,LsExp)]
        ;
        Cabeza == 40, !,
        NumNuevo is Num+1,
        append(LsCarExa,[Cabeza],LsCarExaParc),
        detec_formula_compleja(Formula,Resto,NumNuevo,LsCarExaParc,SbLsFormula)
        ;
        Cabeza == 41, !,
        NumNuevo is Num-1,
        append(LsCarExa,[Cabeza],LsCarExaParc),
        detec_formula_compleja(Formula,Resto,NumNuevo,LsCarExaParc,SbLsFormula)
        ;
        append(LsCarExa,[Cabeza],LsCarExaParc),
        detec_formula_compleja(Formula,Resto,Num,LsCarExaParc,SbLsFormula).



mensaje_error(Formula):-
        nl,
        write('La consulta esta mal formulada, no ajustandose a las normas'),nl,
        write('Existe un error de sintaxis en '), write(Formula),nl,
        write('Por favor, vuelva a iniciar el proceso corrigiendo el error detectado'),
        nl.



prc_compr_exp(LsOperac,LsExp):-
        LsExp == [], !,
        calc_operac_pend(LsOperac)
        ;
        LsExp = [Cabeza|Cola],
        name(Cabeza,LsCarCabeza),
        LsCarCabeza = [34|_], !,
        prc_compr_exp(LsOperac,Cola)
        ;
```

```
        LsExp = [Cabeza|Cola],
        sb_prc_interrog(Cabeza,LsOperac,Cola).
lct_y_abd_todo_arch(Arch):-
        open(Arch,read,E),
        repeat,
        read(E,X),
        (
        X == end_of_file, !,
        close(E)
        ;
        assertz(X),
        fail
        ).



calc_operac(SbLsFormula):-
        SbLsFormula = [_,prc(_,LsExp)],
        LsSetExp = [],
        sb_prc_calc_operac(SbLsFormula,LsExp,LsSetExp).



sb_prc_calc_operac(SbLsFormula,LsExp,LsSetExp):-
        LsExp == [], !,
        realizac_operac(SbLsFormula,LsSetExp)
        ;
        LsExp = [Cabeza|Cola],
        name(Cabeza,LsCarCabeza),
        LsCarCabeza = [34|Resto], !,
        append(LsCarPalabra,[34],Resto),
        name(Palabra,LsCarPalabra),
        (
        pl(_,Palabra,_,_,LsOcurrPal), !,
        findall(NDPal,member([NDPal,_,_,_],LsOcurrPal),LsNDPal),
        list_to_set(LsNDPal,SetPal)
        ;
        SetPal = []
        ),
        append(LsSetExp,[SetPal],LsSetExpAct),
        sb_prc_calc_operac(SbLsFormula,Cola,LsSetExpAct)
        ;
        LsExp = [Cabeza|Cola],
        formula(Cabeza,_,_,SetRsl),
        append(LsSetExp,[SetRsl],LsSetExpAct),
        sb_prc_calc_operac(SbLsFormula,Cola,LsSetExpAct).
```

```
realizac_operac(SbLsFormula,LsSetExp):-
        SbLsFormula = [Formula,prc(Operador,LsExp)],
        prc_realizac_operac(Operador,LsSetExp,SetRsl),
        assertz(formula(Formula,prc(Operador,LsExp),LsSetExp,SetRsl)).
prc_realizac_operac(y,[Seta,Setb],SetRsl):-
        intersection(Seta,Setb,SetInt),
        sort(SetInt,SetRsl).


prc_realizac_operac(o,[Seta,Setb],SetRsl):-
        union(Seta,Setb,SetUni),
        sort(SetUni,SetRsl).


prc_realizac_operac(xor,[Seta,Setb],SetRsl):-
        intersection(Seta,Setb,SetInt),
        union(Seta,Setb,SetUni),
        subtract(SetUni,SetInt,SetXor),
        sort(SetXor,SetRsl).


prc_realizac_operac(no,[Seta],SetRsl):-
        open('C:/Mis documentos/prolog/datoscol.txt',read,E),
        read(E,LsDocsCol),
        close(E), !,
        subtract(LsDocsCol,Seta,SetRsl).


escribe_lista([]):- !.

escribe_lista([Cabeza|Cola]):-
        write(Cabeza),tab(2),
        escribe_lista(Cola).
```

## MÓDULO DE INTERROGACIÓN EN LENGUAJE NATURAL

- Objetivo: Realización de una consulta en lenguaje natural y vectorización de la misma.

## MÓDULO DE CÁLCULO DE SIMILARIDAD

- Objetivo: cálculo de la similaridad entre dos vectores (documentos o documento y consulta) con tres modalidades optativas: coeficientes del Coseno, Dice y Jaccard.
- Característica destacada: una vez realizada una modalidad de similaridad es posible realizar otra posterior sin necesidad de reiniciar el sistema.

## MÓDULO DE CÁLCULO DE CLUSTERS MEDIANTE ENLACE SIMPLE

- Objetivo: ordenación de los documentos de la colección conforme a su grado de semejanza con la consulta mediante técnica de clusters por enlace simple.
- Característica destacada: efectúa vectorización de la consulta y de la colección mediante tf.idf y calcula la similaridad entre consulta y centroide de cluster anterior mediante Coeficiente del Coseno.