Gestor de documents

Identificador de l'equip: 11.1

Pol Camprubí Prats: pol.camprubi.prats@estudiantat.upc.edu
Pol Mañé Roiger: pol.mane@estudiantat.upc.edu
Isaac Roma Granado: isaac.roma.granado@estudiantat.upc.edu
Juli Serra Balaguer: juli.serra@estudiantat.upc.edu

Versió de lliurament 1.1
1ª Entrega PROP
14/11/2022

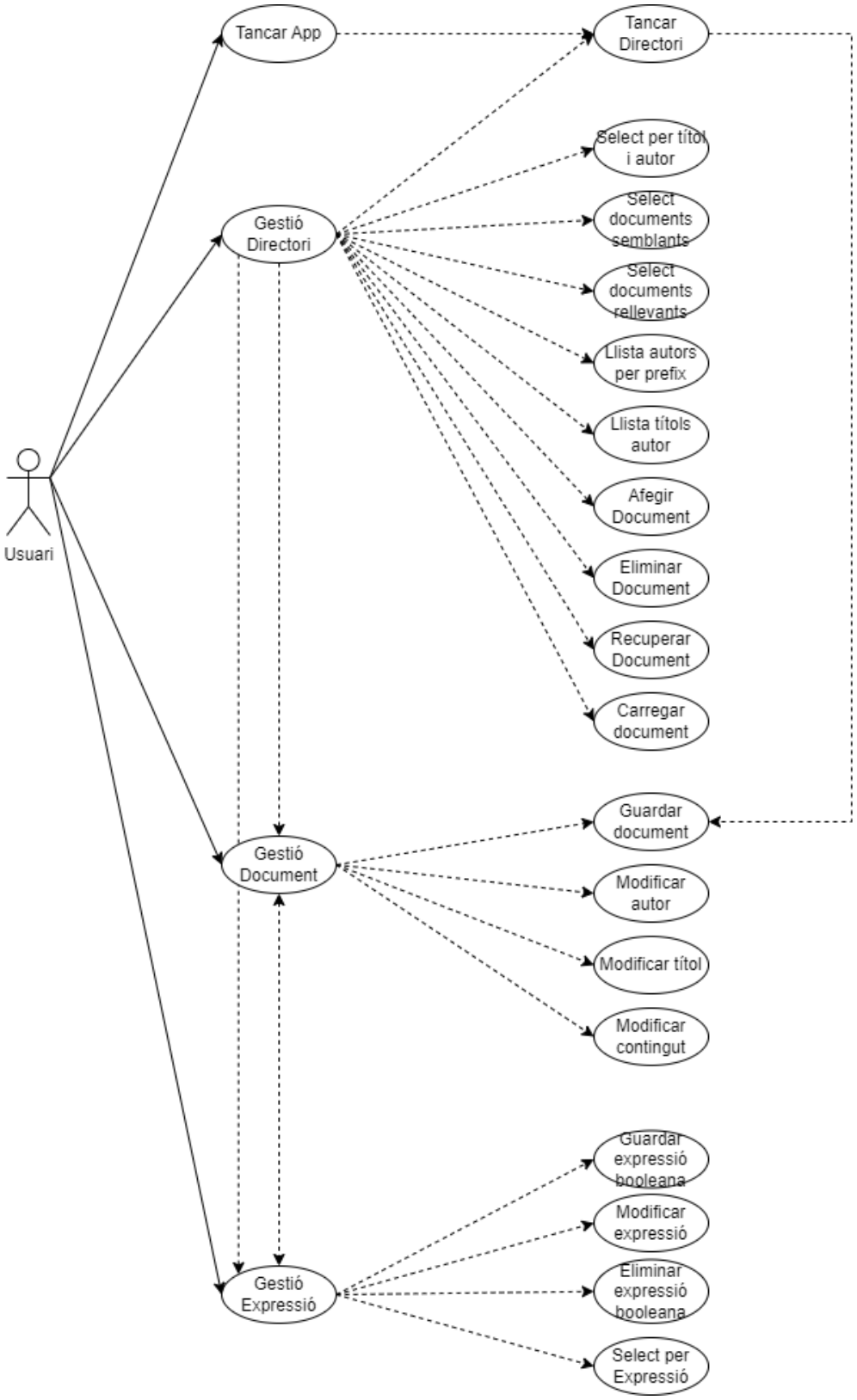
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Índex

1. Diagrama de casos d'ús

A continuació mostrem el diagrama de casos dús. El diagrama es troba detallat al directori de Google Drive.



1.1 Descripció de casos d'ús

Nom: Tancar Aplicació

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari farà clic al botó de tancar l'aplicació
2. El sistema seguirà el procediment de Tancar Directori
3. El sistema termina l'aplicació

Errors possibles i cursos alternatius:

1. No hi ha un directori obert, el sistema termina l'aplicació(salta al pas 3)

Nom: Tancar directori

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica que vol tancar el directori obert.
2. Si el directori obert té canvis que no s'han guardat, el sistema mostrarà a l'usuari una finestra amb 3 opcions
    a. Guardar el Document. Segueix al punt 4
    b. Tancar Document sense guardar-lo. Segueix al punt 3
    c. Cancel·lar (surt del cas d'ús)
3. Si s'ha seleccionat la opció b), notificarà a l'usuari que està a punt de tancar el document sense guardar-lo amb dues opcions.
    a. Sí, segueix al punt 4.
    b. No, torna al punt 2.
4. El sistema tanca el document obert i torna a "l'estat inicial" de l'aplicació

Errors possibles i cursos alternatius:

1. No hi ha un directori obert (no fa res/no surt la opció).

Nom: Select contingut per nom d'autor i títol

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol fer una cerca fent click al botó on posa buscar per nom d'autor i títol
2. El sistema mostra una finestra amb les possibles cerques que pot realitzat
3. L'usuari clica en l'opció de cerca per nom d'autor i títol
4. El sistema mostra una nova finestra amb dos camps de text, un per l'autor i l'altra pel títol
5. L'usuari escriu el nom d'autor i el títol pel què vol realitzar l'operació
6. El sistema mostra el documents on els atributs autor i títol són el mateix que els valors que ha ficat l'usuari en els camps de text

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment

Nom: Select documents semblants

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indiaca al sistema que vol fer una cerca
    a. Ex: Fer click dret dins el directori
2. El sistema mostra una finestra amb les possibles cerques que pot realitzat
3. L'usuari clica en l'opció de cerca per documents semblants
4. El sistema mostra una nova finestra amb un botó i un camp de text, el botó per tal de pujar un document i el camp de text per inserir un integer "x"
5. L'usuari carrega el document a comparar i introdueix l'integer per poder realitzar l'operació
6. El sistema mostra els "x" documents més semblants al document utilitzat per fer la consulta

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment
2. El document pujat no està en un dels formats definits
3. El valor de l'integer inserit no és un integer

Nom: Select per expressió

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol fer una cerca
    a. Ex: Fer click dret dins el directori
2. El sistema mostra una finestra amb les possibles cerques que pot realitzar
3. L'usuari clica en l'opció de cerca per expressió.
4. El sistema mostra una nova finestra amb:
    a. un camp de text i un "checkbox"(seguir a pas 6)
    b. un desplegable amb les expressions guardades(seguir a pas 5)
5. Si l'usuari selecciona una expressió ja guardada el sistema continua al pas
6. L'usuari escriu l'expressió booleana i els conjunt de paraules amb els quals vol realitzar la cerca i pot seleccionar o no el checkbox
7. Si la checkbox és seleccionada, el sistema segueix el cas d'ús guarda expressió booleana a partir del punt 6.
8. El sistema mostra els tots els documents que contenen una frase que satisfà l'operació booleana introduïda.

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment
2. El valor del string és NULL
3. L'expressió no és declarada correctament

Nom: Select documents rellevants

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol fer una cerca
    a. Ex: Fer click dret dins el directori
2. El sistema mostra una finestra amb les possibles cerques que pot realitzar
3. L'usuari clica en l'opció de cerca documents rellevants.
4. El sistema mostra una nova finestra amb dos camps de text.
5. L'usuari introdueix una serie de paraules p en el primer camp de text i un enter k en el segon.
6. El sistema mostra els k documents més rellevants segons el llistat de paraules introduïdes.

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment
2. El valor de l'integer introduït no és un integer

Nom: Llista d'autors per prefix

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol fer una cerca
    a. Ex: Fer click dret dins el directori
2. El sistema mostra una finestra amb les possibles cerques que pot realitzar
3. L'usuari clica en l'opció de cerca autors per prefix
4. El sistema mostra una nova finestra amb un camp de text.
5. L'usuari introdueix el prefix amb el qual pot fer la cerca.
6. El sistema mostra una llista amb tots els autors que comencen amb el prefix introduït.

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment
2. El valor del string no és un string

Nom: Llista de títols d'un autor

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol fer una cerca
    a. Ex: Fer click dret dins el directori
2. El sistema mostra una finestra amb les possibles cerques que pot realitzar
3. L'usuari clica en l'opció de cerca els títols d'un autor
4. El sistema mostra una nova finestra amb un camp de text.
5. L'usuari introdueix el nom de l'autor del qual ven vol obtenir els títols.

6. El sistema mostra una llista amb tots els títols de l'autor introduït.

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment
2. El valor del string no és un string

Nom: Afegir document

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol crear un document fent click en el botó de crear
2. El sistema mostra una finestra amb un document en blanc on l'usuari podrà afegir l'autor, el títol i el contingut del document
3. El sistema segueix el cas d'ús de "Guardar Document" a partir del pas 2

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment
2. Ja existeix un document amb autor i títol igual
3. El format del document no és correcte

Nom: Eliminar document

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol eliminar un document
2. El sistema informa a l'usuari si n'està segur
    a. Esborrar, continua al pas 3
    b. Cancel·la, surt del cas d'ús
3. El sistema esborra el document del directori i modifica la taula de pesos, la cua i els vectors corresponents
4. El sistema elimina el document

Errors possibles i cursos alternatius:

Nom: Recuperar document

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol recuperar un document fent click al botó on posa obrir
2. El sistema mostra una finestra amb un camp de text on l'usuari indicarà l'autor i el títol del document a recuperar.
3. El sistema mostra una finestra amb el document seleccionat per l'usuari

Errors possibles i cursos alternatius:

1. Que el document no existeix en el disc dur

Nom: Carregar document

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari navega pel directori i indica al sistema que vol carregar un document
    a. Ex: Fer doble click sobre el document
2. El sistema s'encarrega d'obrir el document en qüestió

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol
2. L'usuari ja té un document obert

Nom: Guardar document

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica que vol guardar un document actual
    a. Ex: "Ctrl + S"
2. El sistema afegeix el document en el directori obert i li assigna una nova id. Per defecte es guardarà amb el format .prop
3. El sistema crea un nou document amb els atributs desitjats per l'usuari

Errors possibles i cursos alternatius:

4. Ja existeix un document amb autor i títol
5. No hi ha un document obert

Nom: Modificar autor

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica al sistema que vol modificar el document
2. El sistema mostra una finestra amb el document
3. L'usuari escriu el nou nom de l'autor del document
4. El sistema modifica l'autor del document seleccionat pel nou autor que ha escrit l'usuari

Errors possibles i cursos alternatius:

1. L'usuari pot avortar/desfer el canvi en qualsevol moment
2. El nou autor és una cadena buida
3. El nou autor és el mateix que el d'un altre document amb el mateix títol

Nom: Modificar títol

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L’usuari indica al sistema que vol modificar el document
2. El sistema mostra una finestra amb el document
3. L’usuari escriu el nou títol del document
4. El sistema modifica el títol del document seleccionat pel nou títol que ha escrit l’usuari

Errors possibles i cursos alternatius:

1. L’usuari pot avortar/desfer el canvi en qualsevol moment
2. El nou títol és una cadena buida
3. El nou títol és el mateix que el d’un altre document amb el mateix autor

Nom: Modificar contingut

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L’usuari indica al sistema que vol modificar un document
2. El sistema mostra una finestra amb el document
3. L’usuari escriu el nou contingut del document
4. El sistema modifica el contingut del document seleccionat pel nou contingut que ha escrit l’usuari i calcula la matriu de pesos

Errors possibles i cursos alternatius:

1. L’usuari pot avortar/desfer el canvi en qualsevol moment
2. El nou contingut conté paraules noves que no existien a tots els documents del sistema -> S'haurà de recalcular el mapa de pesos de tots els documents i el map de pesos de la classe Directori

Nom: Guardar expressió booleana

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L’usuari indica que vol guardar un l’expressió booleana
    a. Ex: Fer clic en dret dins el directori
2. El sistema mostra una finestra amb els possibles canvis que pot realitzar
3. L’usuari clica en l’opció de guardar expressió booleana
4. El sistema mostra una nova finestra amb un camp de text.
5. L’usuari introdueix l’expressió booleana a guardar.
6. El sistema li assigna una id a l'expressió i la guarda en el sistema

Errors possibles i cursos alternatius:

1. L’usuari pot avortar l’operació en qualsevol moment

Nom: Modificar expressió booleana

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L’usuari indica al sistema que vol fer un canvi

    a. Ex: Fer click dret sobre el document
2. El sistema mostra una finestra amb els possibles canvis que pot realitzar
3. L'usuari clica en l'opció de modificar expressió booleana
4. El sistema mostra una nova finestra amb les expressions guardades
5. L'usuari selecciona la que desitja modificar
6. El sistema mostra una nova finestra amb un camp de text amb l'expressió booleana seleccionada
7. L'usuari modifica l'expressió
8. El sistema canvia l'expressió

Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment
2. El valor de l'integer inserit no és un integer

Nom: Eliminar expressió booleana

Actor: Usuari

Comportament (diàleg entre els actors i el sistema):

1. L'usuari indica que vol eliminar una expressió
    a. Ex: Fer clic en dret dins el directori
2. El sistema mostra una finestra amb els possibles canvis que pot realitzar
3. L'usuari clica en l'opció d'eliminar expressió booleana
4. El sistema mostra les expressions booleanes guardades
5. L'usuari selecciona l'expressió a eliminar
    a. Ex: L'usuari clica la "X" al costat de l'expressió
6. El sistema mostra un avís de confirmació d'eliminació amb dos opcions a escollir per l'usuari
    a. Esborrar (salta al punt 7)
    b. Cancel·la (salta al punt 4)
7. El sistema elimina l'expressió booleana

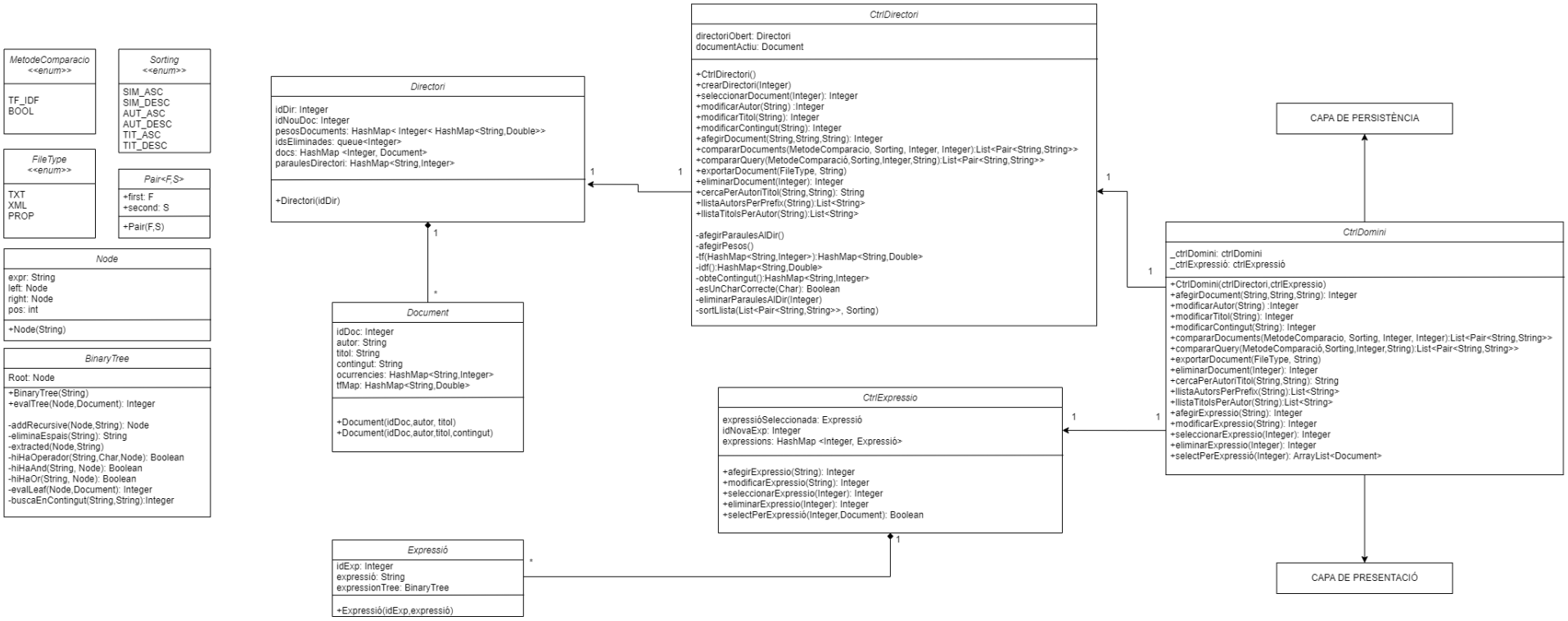
Errors possibles i cursos alternatius:

1. L'usuari pot avortar en qualsevol moment (surt del cas d'ús)

2. Diagrama del model conceptual

2.1 Disseny del diagrama de model conceptual

A continuació mostrem el diagrama del model conceptual. El diagrama es pot trobar detallat al directori de Google Drive i en la carpeta DOCS d'aquest repositori.



Restriccions textuals:

-Claus primàries:

(**Directori,** Integer: idDir)

(**Document,** Integer: idDoc)

(**Expressió**, Integer: idExp)

-Claus Candidates:

(**Document,** autor + títol)

2.2 Descripció de les classes

Directori: La classe Directori és aquella que conté l'agrupació de tots els Documents de l'usuari. S'encarrega principalment de tenir guardats i ubicats els documents i guardar la informació corresponent de cada un per tal de que el CtrlDirectori pugui realitzar les operacions sense problema.

Document: La classe Document representa un dels documents compresos dins el directori. Aquesta classe té un atribut privat i tres de públics. L'atribut privat és la id (PK de la classe) que serveix al sistema per identificar el document que l'usuari està intentant accedir en aquell moment. Els atributs públics són: autor, títol i contingut. Com hem dit anteriorment, autor i títol formen una clau candidata ja que no ens interessa tenir guardat un document on aquests dos atributs siguin el mateix.

Expressió: La classe Expressió guarda tota la informació necessària per tal de fer les operacions sobre ella. En aquest cas tenim una expressió que representa un String, un atribut id que fa la mateixa funció que en la classe Document i un expression Tree necessari per tal de calcular quins documents compleixen l'expressió.

Arribats a aquest punt, ja tenim descrites totes les classes del nostre sistema. Fixem-nos que cap classe té operacions importants, només setters, getters i creadores. A continuació, veurem els controladors del nostre sistema que són els encarregats d'executar els mètodes necessaris per fer les operacions corresponents:

CtrlDomini: El controlador de Domini s'encarrega d'instanciar la resta de controladors de la capa de domini i proporciona una interfície més pròxima a l'usuari dels mètodes de la resta de controladors com el CtrlDirectori o CtrlDocument.

CtrlDirectori: El controlador de directori s'encarrega de la gestió dels mètodes de les classes Directori i Document. Es comunica amb Directori per tal de realitzar totes les funcionalitats referents a aquests. Els atributs d'aquesta classe

són el directori obert que s'està utilitzant i el document que té obert en aquell moment l'usuari

CtrlExpressió: El controlador d'Expressions s'encarrega de gestionar els mètodes guardar, modificar, eliminar i comparar els documents del directori amb les expressions que l'usuari ha afegit anteriorment.

Finalment, descrivim les classes complementàries de l'UML:

Pair<F,S>: La classe Pair és una classe genèrica per tal de crear tuples de dos elements. L'ús del pair és per aquelles operacions que s'esperi el retorn de dos variables bàsiques (Strings. Integers, Booleans,...).

Node: La classe node és creada per tal de fer la classe BinaryTree que explicarem a continuació. Un node té els atributs següents: un string que representa la expressió, un integer que recorre l'expressió com ho faria un punter i dos nodes corresponents al fill esquerra i el fill dret.

BinaryTree: La classe BinaryTree és l'encarregada d'estudiar l'expressió que ha ofert l'usuari, convertir-la en un arbre binari segons unes instruccions donades i ser capaç d'avaluar la funció resultant de la manera més simple possible. Tota aquesta classe està implementada utilitzant recursivitat.

2.3 Descripció de atributs i mètodes de cada classe

Totes les operacions que poden tenir una excepció (aquelles que tenen return int) retornen un dels següents resultats:

- 10 si l'operació ha funcionat correctament.
- 11 s'executa correctament però avisa que l'objecte eliminat estava en ús per l'usuari
- 20 si l'operació té un error per objecte ja existent
- 30 si l'operació té un error per un valor null o paràmetre no vàlid.
- 31 si l'operació referencia a un objecte null

Classe Document

int idDoc
Representa el número identificador d'un document.

String autor
Representa l'autor d'un document.

String títol
Representa el títol d'un document.

String contingut
Representa el contingut d'un document.

HashMap<String, Integer> ocurrencies
Representa les vegades que apareix cada paraula en el contingut del document.

HashMap<String, Double> tfMap
Representa el càlcul de l'operació tf per aquell document

public Document(int idDoc, String autor, String titol)
Constructora de document.

public Document(int idDoc, String autor, String titol, String contingut)
Constructora de document.

Classe Directori

int iDir
Representa el número identificador d'un directori.

HashMap<Integer, HashMap<String, Double>> pesosDocs
Representa la taula de pesos del nostre sistema.

Queue<Integer> deletedIds
Representa els números identificadors dels documents eliminats.

int idNouDoc
Representa un contador per atribuir la id corresponent al nou document.

HashMap<Integer, Document> docs
Representa els documents que existeixen en el directori.

HashMap<String, Integer> paraulesDirectori
Representa les paraules que trobem en el directori seguit de les ocurrències.

public Directori(int idDir)
Constructora de directori.

Classe Expressió

int idExp
Representa el número identificador d'una expressió.

String expressio
Representa una expressió.

BinaryTree ExpressionTree
Representa l'arbre binari d'una expressió.

public Expressio(Integer idExp, String expressio)
Constructora d'expressió.

Classe CtrlDirectori

Directori directoriObert
Representa el directori que s'està utilitzant.

Document documentActiu
Representa el document que l'usuari està usant

public CtrlDirectori()
Constructora de controlador de directori.

public void crearDirectori(int idDir)
Operació per crear un directori en el nostre sistema.

public int seleccionarDocument(int idDoc)
Operació per obrir un document que ja teníem carregat dins el nostre sistema.
@param idDoc és l'identificador del document que volem obrir.

public int modificarAutor(String autor)
Operació per modificar l'autor d'un document.
@param autor és el nou nom d'autor que es vol utilitzar pel document.

public int modificarTitol(String titol)
Operació per modificar el títol d'un document.
@param títol és el nou nom del títol que es vol utilitzar pel document.

public int modificarContingut(String contingut)
Operació per modificar el contingut d'un document.
@param contingut és el nou contingut que es vol utilitzar pel document.
A part de modificar el contingut es realitzen les següents operacions:

- S'actualitza el atribut paraulesDirectori restant 1 a les paraules que contenia el document

- Inicialitzem mb el valor corresponent l'atribut ocurrencies del document modificat.
- Inicialitzem amb el valor corresponent l'atribut tfMap del document modificat.
- Afegim les paraules del nou contingut a l'atribut paraulesDirectori.
- Afegim els pesos corresponents a l'atribut pesosDocs.

public int afegirDocument (String autor, String titol, String contingut)
Operació per donar d'alta un document en el directori.
@param autor representa l'autor del document que es vol afegir.
@param títol representa el títol del document que es vol afegir.
@param contingut representa el contingut del nou document.
A part de crear el Document es realitzen les següents operacions:
- S'afegeix el document a l'atribut docs del directoriObert
- Inicialitzem amb el valor corresponent l'atribut ocurrencies de document creat
- Inicialitzem amb el valor corresponent l'atribut tfMap del document creat
- Afegim les paraules del document a l'atribut ParaulesDirectori
- Afegim els pesos corresponents a l'atribut pesosDocs

private void afegeixParaulesAlDir()
Operació per afegir noves parules a ParaulesDirectori cada cop que es crea un document nou o que es modifica el contingut d'un document ja existent

private void afegeixPesos()
Operació per calcular i afegir els pesos de cada paraula en el directori.

private HashMap<String, Double> tf(HashMap<String, Integer> paraules)
Operació per calcular el tf d'un document concret
@param paraules representa el HashMap on es guarden les ocurrències de cada paraula en el document.

private HashMap<String, Double> idf()
Operació per calcular l'idf en el nostre sistema.

private HashMap<String, Integer> obteContingut()
Operació que serveix per obtenir el contingut del document actiu.

private boolean esUnCharCorrecte(char c)
Operació que serveix per saber si un caràcter compleix els requisits per ser considerat vàlid
@param c representa el caràcter que volem evaluar.

public enum METODE_COMPARACIO

Enumeració dels diferents tipus de comparació que podem realitzar (TF_IDF i BOOL).

public enum SORTING
Enumeració de les possibles ordenacions al comparar documents (similaritat ascendent, similaritat descendent, autors ascendent, autors descendent, títol ascendent i títol descendent).

public List<Pair<String, String>> compararDocuments(METODE_COMPARACIO m, SORTING s, Integer k, Integer IdDoc)
Operació que serveix per comparar un document amb els que tenim en el directori i obtenir-ne els k més semblants. A part, el resultat el podem obtenir mitjançant dos mètodes diferents, el TF-IDF i el BOOLEÀ (a decidir per l'usuari) i podem ordenar aquest resultat segons els criteris que ens ofereix el SORTING (similaritat ascendent, similaritat descendent, autors ascendent, autors descendent, títol ascendent i títol descendent).
@param m representa els diferents tipus de comparació que podem realitzar.
@param s representa el criteri d'ordenació pel resultat.
@param k representa el nombre de documents semblants que volem obtenir.
@param IdDoc representa el número identificador d'un document.

private void sortLlista(List<Pair<String, String>> llistaSemblants, SORTING s)
Operació que serveix per ordenar els documents semblants segons els diferents criteris d'ordenació que ens ofereix el SORTING (similaritat ascendent, similaritat descendent, autors ascendent, autors descendent, títol ascendent i títol descendent).
@param llistaSemblant representa la llista que es vol ordenar.
@param s representa el criteri d'ordenació pel resultat.

public List<Pair<String, String>> compararQuery(METODE_COMPARACIO m, SORTING s, Integer k, String paraules)
Operació que serveix per obtenir els documents més rellevants pel que fa a contingut segons la query de paraules passada com a paràmetre. Aquest mètode crea un nou document dins el sistema (que és eliminat un cop s'acaba l'operació) i té fa les mateixes funcionalitats que l'operació compararDocuments.
@param m representa els diferents tipus de comparació que podem realitzar.
@param s representa el criteri d'ordenació pel resultat.
@param k representa el nombre de documents rellevants que volem obtenir.
@param paraules representa la query que l'usuari entra com a variable per examinar.

public enum FILETYPE
Mètode que enumera els diferents tipus de formats en els que podem tenir un document (TXT i XML).

public void exportarDocument(FILETYPE format, String path)
Operació per exportar el document del directori a un path elegit.
@param format es correspon en quin format es desitja exportar el document.
@param path es correspon al camí desitjat per tal de guardar el document.

public int eliminarDocument(int idDoc)
Operació per eliminar un document del directori.
@param idDoc és l'identificador del document que es vol eliminar del directori.
Abans d'eliminar el document, el sistema:

- Resta les paraules del document a l'atribut paraulesDirectori
- Elimina el document de la taula de pesos (PesosDocs)
- Recalcula els pesos amb els documents que queden
- Elimina el document de l'atribut docs
- Afegeix la id del document borrat a la cua de ids

private void eliminarParaulesAlDir(int idDoc)
Operació per eliminar les paraules del document idDoc en el directori.
@param idDoc representa el número identificador del document que volem eliminar.

public String cercaPerAutoriTitol(String autor, String titol)
Operació per cercar un document del directori segons títol i autor.
@param autor representa l'autor del document que estem cercant.
@param títol representa el títol del document que estem cercant.

public List<String> llistaAutorsPerPrefix(String pre, SORTING s)
Operació que retorna una llista amb els documents que tenen un autor que comença amb el prefix passat com a paràmetre.
@param pre representa el prefix pel qual volem realitzar la cerca.
@param s representa el criteri d'ordenació pel resultat.
El resultat es pot ordenar segons les dues opcions de SORTING ofertes a l'usuari (ordenació per nom d'autors ascendent i descendent).

public List<String> llistaTitolsPerAutor(String autor, SORTING s)
Operació que retorna una llista amb els documents que tenen com a autor l'autor que es passa per paràmetre.
@param autor representa l'autor pel qual volem fer la cerca.
@param s representa el criteri d'ordenació pel resultat.
El resultat es pot ordenar segons les dues opcions de SORTING ofertes a l'usuari (ordenació per nom d'autors ascendent i descendent).

Classe CtrlDomini

CtrlDirectori _ctrlDirectori

Representa el controlador de directori.

CtrlExpressio _ctrlExpressio
Representa el controlador d'expressió.

public CtrlDomini(CtrlDirectori _ctrlDirectori, CtrlExpressio _ctrlExpressio)
Creadora del controlador de domini
@param _ctrlDirectori representa el controlador de directori.
@param _ctrlExpressió representa el controlador d'expressió.

public ArrayList<Document> selectPerExpressio(Integer idExp)
Operació que serveix per obtenir els document que compleixin l'expressió passada com a paràmetre. El controlador consegueix l'atribut docs de controlador de directori. Un cop té el valor de docs, itera sobre tots els documents del directori obert i crida a la funció selectPerExpressio de la classe CtrlExpressio per comprovar si compleix o no l'expressió passada com a paràmetre.
@param idExp representa el número identificador de l'expressió la qual es vol comprovar.

Els altres mètodes d'aquesta classe no varien de la seva descripció en la classe corresponent.

Classe CtrlExpressio

Expressio expressioSeleccionada
Representa l'expressió sobre la que es treballa.

Integer IdNovaExp
Representa el número identificador de la nova expressió.

HashMap<Integer,Expressio> expressions
Representa les expressions que hi han en el directori.

public CtrlExpressio()
Constructora de CtrlExpressio.

public int seleccionarExpressio (Integer idExp)
Operació que serveix per seleccionar l'expressió seleccionada dins el sistema
@param idExp representa el número identificador de l'expressió que volem seleccionar.

public int afegirExpressio(String expressio)

Operació que serveix per afegir una expressió al directori. L'operació actualitza l'atribut expressions de la classe CtrlExpressio afegint la nova expressió si no existia ja en el directori.
@param expressió representa l'string que l'usuari reconeix com l'expressió per fer la cerca.

public int modificarExpressio(String exp)
Operació que serveix per modificar l'expressió seleccionada amb un nou text corresponent. Es modifica l'expressió sempre que no existeixi ja una expressió en el directori amb el mateix valor que l'expressió passada com a paràmetre.
Si es modifica l'expressió, es crea un nou BinaryTree de la nova expressió.
@param exp representa la nova expressió.

public int eliminarExpressio(int idExp)
Operació que serveix per eliminar l'expressió amb idExp del sistema.
@param idExp representa la id de l'expressió que volem eliminar.

public boolean selectPerExpressio(Integer idExp, Document document)
Operació que serveix per buscar, de tots els documents que nosaltres tenim en el sistema, aquells que compleixen l'expressió passada com a paràmetre.
Retorna una llista amb els documents que compleixen l'expressió.
Es crida a l'operació evalTree de la classe BinaryTree per calcular el resultat.
@param idExp representa el número identificador de l'expressió que volem avaluar.
@param document representa el document possible a avaluar.

Classe BinaryTree

class Node:
Representa un node del BinaryTree.

Node root
Representa el node arrel.

public BinaryTree(String exp)
Creadora de la classe BinaryTree.
Crea un BinaryTree de l'expressió que li entra com a paràmetre.
@param exp representa l'expressió sobre la que es vol crear el BinaryTree.

private Node addRecursive(Node current, String exp)
Operació que serveix per crear l'arbre binari de manera recursiva. El codi té com a prioritats els parèntesis, les ands i, finalment, les ors.
@param current representa el node en que volem treballar.
@param exp representa l'expressió que estem utilitzant.

private String eliminaEspais(String exp)
Operació que serveix per eliminar els espais d'una expressió.
Per tal de eliminar els espai es crida a la funció isWhitespace per determinar si tenim un espai en blanc o no.
@param exp representa l'expressió de la qual es volen eliminar els espais.

private void extracted(Node current, String exp)
Operació que serveix per avaluar l'expressió dividint-la en dues parts tenint com a node actual el operador de l'expressió.
@param current representa el node en que volem treballar.
@param exp representa l'expressió que estem utilitzant.

private boolean hiHaOperador(String exp, char op, Node current)
Operació que serveix per comprovar si una expressió conté un operador.
@param exp representa l'expressió que estem utilitzant.
@param op representa l'operador que estem buscant.
@param current representa el node en que volem treballar.

private boolean hiHaAnd(String exp, Node current)
Operació que serveix per comprovar si hi ha l'operador AND en l'expressió passada com a paràmetre. Per tal de fer la comprovació es crida a la funció hiHaOperador.
@param exp representa l'expressió que estem utilitzant.
@param current representa el node en que volem treballar.

private boolean hiHaOr(String exp, Node current)
Operació que serveix per comprovar si hi ha l'operador OR en l'expressió passada com a paràmetre. Per tal de fer la comprovació es crida a la funció hiHaOperador.
@param exp representa l'expressió que estem utilitzant.
@param current representa el node en que volem treballar.

public static int evalTree(Node current, Document d)
Operació que ens serveix per avaluar el tree de manera recursiva. El current és el node en que estem treballant i el document representa l'objecte que nosaltres estem passant a la funció per veure si compleix l'expressió.
@param current representa el node en que volem treballar.
@param d representa el document que estem comparant.

public static int evalLeaf(Node current, Document d)
Quan la funció evalTree troba que el node és una fulla, crida al mètode evalLeaf que analitza i calcula cada fulla de l'arbre.
@param current representa el node en que volem treballar.
@param d representa el document que estem comparant.

private static int buscaEnContingut(String word, String contingut)
Operació que serveix per comprovar si una paraula apareix en el contingut d'un document.
@param word representa la paraula que volem cercar.
@param contingut representa el contingut del document en el que volem fer la cerca.

Classe Pair

F _first
Representa el primer atribut de la classe Pair.

S _second
Representa el segon atribut de la classe Pair.

public Pair(F first, S second)
Constructora de la classe Pair
@param first representa el primer atribut de la classe Pair.
@param second representa el segon atribut de la classe Pair.

public boolean equals(Object o)
Operació que sobrescriu la funció equals i serveix per comprovar si un pair és igual a un altre pair pel que fa al contingut.
@param o representa el atribut a comparar.

public String toString()
Operació que serveix per passar els objectes de la classe Pair a string.

3.Repartició de feina

En aquest document es pretén explicar la part de la primera entrega que ha dut a terme cada integrant de l'equip. La repartició ha estat negociada i acceptada entre tots els membres.

1. Documentació

    - Diagrama de casos d'ús: Tots
    - Model Conceptual: Tots
    - Testeig mètodes de les classes: Pol Mañé
    - Document 1a Entrega: Isaac, Pol Camprubí i Pol Mañé

2. Capa de domini

    Classes:

    - Directori: Juli
    - Document: Pol Camprubí
    - Expressió: Isaac
    - Pair: Pol Camprubí
    - BinaryTree: Pol Camprubí

    Controladors:

    - CtrlDomini: Pol Camprubí
    - CtrlDirectori: Juli i Pol Camprubí
    - CtrlExpressió: Isaac i Pol Camprubí

    Drivers

    Pol Mañé

Per tal de veure qui ha fet què en els documents que s'han creat compartits, es pot observar en la IDE on, a sobre el nom de cada operació, hi ha el nom de la persona que hi ha estat treballant.

4.Estructura de dades i algorismes utilitzats

4.1 Estructures de dades

4.1.1 ArrayList

Una classe ArrayList és una matriu redimensionable, que està present a Java. Tot i que les matrius integrades tenen una mida fixa, les ArrayLists poden canviar la seva mida de forma dinàmica. Els elements es poden afegir i eliminar d'una ArrayList sempre que calgui, ajudant l'usuari amb la gestió de la memòria.

L'avantatge que té respecte a els vectors és que quan augmenta la dimensió de l'estructura de dades, el vector dobla la seva capacitat mentre que l'arrayList només demana el 50%.

Els temps d'inserció i eliminació és constant. Per trobar un element, com que no està ordenat de cap manera, el temps és **O(n)**.

L'ús de les ArrayList en aquest projecte ha estat per donar resultats i com a utilització per tal de guardar informació en mètodes específics.

4.1.2 Diccionaris

Per tal de representar una taula de pesos dins el nostre sistema hem triat per l'opció dels diccionaris. L'altra opció que hi havia sobre la taula era una matriu de vectors, era una possibilitat molt viable a l'hora de programar, ja que l'espai en memòria que s'usava era mínim, tot i això, es va acabar descartant donat els alts costos temporals que comportaven.

Aquí és on entren els diccionaris, unes estructures de dades que ocupen més espai en memòria, però que ens permeten fer les operacions més ràpidament.

HashMap

Un HashMap és una estructura de dades que utilitza una funció hash per assignar valors d'identificació, coneguts com a claus, amb els seus valors associats. Conté parells "clau-valor" i permet recuperar el valor a partir de la seva clau.

El principal avantatge respecte a les matrius és que només es guarda aquella informació que hem afegit explícitament, ergo l'espai ocupat serà **Θ(x)**. L'altre avantatge és que la cerca en un diccionari és de cost lineal **Θ(x).**

En el nostre sistema podem veure sis diccionaris, tres a la classe Directori, dos a Document i un a CtrlExpressió.

Els mapes de la classe Directori es declaren de la següent manera:

- pesosDocuments -> Hashmap< Integer, Hashmap< String, Double>> on la key del primer map es correspon a la ID del document i el valor és un altre map. En el segon map la key és una paraula (la qual trobem en el contingut d'algun document dins el nostre sistema) i el value és el pes de la paraula en aquell document.
- docs -> HashMap<Integer, Document> la key d'aquest map representa la id del document que té com a valor. Ens serveix per a guardar tots els documents que tenim en el directori.
- paraulesDirectori -> HashMap<String, Integer> en aquest últim mapa s'hi guarden les paraules que trobem en el directori, on cada paraula representa una entrada en el mapa i el seu valor és el nombre d'ocurrències en el sistema.

Pel que fa a la classe Document tenim els següents mapes:

- ocurrencies -> HashMap <String, Integer> la clau d'aquest mapa són les paraules que tenim dins el contingut del document i el valor és el nombre de vegades que aquella paraula hi apareix. (és com el mapa de paraulesDirectori, però en aquest cas només hi guardem les paraules d'aquell document en especial)
- tfMap -> HashMap<String,Double> aquest mapa es guarda per un motiu d'eficiència, representa el pes de cada paraula del document dins el nostre sistema (la clau és la paraula i el valor el pes).

Finalment, el mapa que trobem a la classe CtrlExpressió és el següent:

- expressions: HashMap <Integer, Expressio> la key d'aquest mapa representa la id de l'expressió que té com a valor. Representen les expressions que nosaltres tenim en el sistema.

El cost per inserció en el map és constant en cas mitjà i lineal. En cas pitjor **O(n)** i el temps d'eliminació té els mateixos costs que els d'inserció així doncs estem davant una de les millors estructura de dades per utilitzar en aquest projecte.

4.1.3 List

La tercera estructura de dades que utilitzem en el projecte són les llistes. De la mateixa manera que les ArrayList, el seu ús en el codi ha estat per donar resultats i com a utilització per tal de guardar informació en mètodes específics.

Com que es tracta de la interfície de la que neix la classe ArrayList, els seus temps d'inserció, eliminació i cerca són els mateixos.

4.1.4 Cua

Finalment, l'última estructura que utilitzem són les cues. Quan eliminem un document dins el nostre directori la seva id quedarà invalidada i no s'usarà més. Per tal de poder "reciclar" les ids implementem aquesta estructura de dades. Gràcies al concepte de FIFO ens permetrà inserir les primeres ids eliminades i que siguin aquestes les primeres a sortir de la cua quan es crea un nou document. Ens podem fixar que una estructura, com per exemple una pila, també ens podria servir.

El cost temporal d'aquesta estructura per inserció i eliminació és immillorable, en ambdós casos és **O(1)**

4.2 Algorismes

4.2.1 Model booleà

El model booleà és un del models més utilitzats per a la recuperació d'informació. Aquest model està basat en la lògica booleana i la teoria de conjunts. Es tracta d'un model binari ja que considera dues possibilitats en relació a un terme en el document: valor 0(f"alse") si aquest és absent, o valor 1("true") si hi és present. No tenim en compte la freqüència dels termes en els documents.

Algunes de les avantatges d'aquest model serien:

1. És un model formal
2. Fàcil d'implementar
3. Model bastant intuïtiu
4. És fàcil identificar els resultats

Però degut a la seva simplesa, tot i el seu gran ús no és dels models més eficaços i presenta diverses desavantatges :

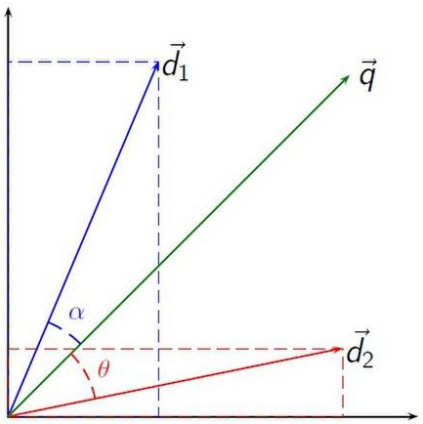
1. Degut a la cerca pot retornar massa o massa pocs documents
2. Tots els termes presents pesen el mateix
3. Com a conseqüència de l'anterior punt, no podem oferir un ordre entre documents que compleixin la cerca

4.2.2 Model vectorial

El model d'espai vectorial és un model algebraic per a representar documents de text com a vectors d'identificadors. Cada document té el seu vector amb el seu pes, si apareix un terme (paraula) en el document el seu valor en el vector és el nombre d'ocurrències en aquell document. S'han desenvolupat diverses formes diferents

de calcular aquests valors, també conegudes com a ponderacions (més endavant explicarem la ponderació tf-idf utilitzada en aquest projecte).

La semblança de documents en una cerca de paraules clau es poden calcular, utilitzant els supòsits de la teoria de similituds de documents, comparant la desviació d'angles entre cada vector de document i el vector de consulta original podem obtenir la similitud entre documents. En la imatge de la dreta podem veure un exemple visual per entendre-ho més clarament, d1 i d2 fan referència als documents avaluats i q representa la consulta. Els angles α i θ representen la semblança.



El model d'espai vectorial té els següents avantatges sobre el model booleà estàndard:

1. Model simple basat en àlgebra lineal
2. Ponderacions de terme no binàries
3. Permet calcular un grau continu de similitud entre consultes i documents.
4. Permet classificar els documents segons la possible rellevància
5. Permet la coincidència parcial

El model d'espai vectorial té les limitacions següents:

1. Els documents llargs estan mal representats perquè tenen valors de similitud deficients (un producte escalar petit i una dimensionalitat gran)
2. Les paraules clau de cerca han de coincidir exactament amb els termes del document; les subcadenes de paraules poden donar com a resultat una "coincidència falsa positiva"
3. Sensibilitat semàntica; els documents amb un context similar, però un vocabulari de termes diferent no s'associaran, cosa que resultarà en una "coincidència falsa negativa".
4. L'ordre en què apareixen els termes al document es perd en la representació de l'espai vectorial.
5. Se suposa teòricament que els termes són estadísticament independents.
6. La ponderació és intuïtiva però no gaire formal.

4.2.3 TF-IDF

En la recuperació d'informació, tf-idf, abreviatura de freqüència de termini-freqüència inversa de documents, és una estadística numèrica que pretén reflectir la importància d'una paraula per a un document. El valor tf-idf augmenta proporcionalment a la quantitat de vegades que una paraula apareix al document i es compensa amb la quantitat de documents al directori que contenen la paraula, la qual cosa ajuda a ajustar el fet que algunes paraules apareixen amb més freqüència en general. Dit amb altres paraules, el pes de cada paraula en el document segueix la fórmula:

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) * \mathrm{idf}(t,D)$$

on

tf(t,d) = en nombre d'ocurrències de t dins el document d

idf(t,D) = el pes de la paraula dins el directori (explicat a continuació)

Seguint la definició anterior, paraules com "el" o "la", que són articles i s'utilitzen molts cops en els documents, tindran un pes inferior que els substantius.

Per saber el pes de la paraula en el directori s'usa la freqüència inversa del document. La freqüència inversa del document és una mesura de quanta informació proporciona la paraula, és a dir, si és comú o rara a tots els documents. És la fracció inversa escalada logarítmicament dels documents que contenen la paraula (obtinguda dividint el nombre total de documents pel nombre de documents que contenen el terme, i després prenent el logaritme d'aquest quocient):

$$\mathrm{idf}(t,D) = \log\left(\frac{N}{|d \in D : t \in d|}\right)$$

on

N: nombre total de documents al directori N = {|D|}

$|d \in D : t \in d|$: nombre de documents d on el terme t apareix dins el directori D.

Tests Unitaris

Els test unitaris estan descrits en el codi a mode de comentari abans de cada mètode.