Projecte de programació - Entrega 3.0

Codi del projecte: 12.2
Codi del clúster: 2.2

Arnau Badía Sampera (*arnau.badia.sampera@est.fib.upc.edu*)
Carla Claverol González (*carla.claverol@est.fib.upc.edu*)
Carlos Lázaro Costa (*carlos.lazaro.costa@est.fib.upc.edu*)
Guillem Castro Olivares (*guillem.castro@est.fib.upc.edu*)

Índex

Manual d'Usuari

Descripció general de l'aplicació

L'aplicació permet realitzar consultes sobre la rellevància entre dades en una xarxa heterogènea. La rellevància entre dues dades es un coeficient entre 0 i 1 que indica el grau de relació i importància relativa entre aquestes.
Aquest coeficient es calcula a partir d'un algoritme per calcular la importància relativa entre entitats en xarxes heterogènees: HeteSim.
L'aplicació permet realitzar gestionar diversos conjunts de dades i fer consultes de la rellevància entre una dada i la resta de dades del conjunt segons una relació (aporta el significat semàntic de la relació entre dades). Es pot afegir un filtre a la consulta, guardar els resultats de les consultes,modificar i filtrar resultats i exportar-los en fitxers de text.

Consultes

Menú principal > Consulta

Una consulta calcula la rellevància entre una dada i la resta de dades del conjunt a partir d'una relació. Quan s'hagin introduit totes les dades desitjades fer click al botó "Consultar" per realitzar la consulta.
Exemple:
Es vol realitzar una consulta sobre la dada de tipus Autor x, amb la relació Autor-Paper-Conferència.
El resultat contindrà en ordre decreixent de rellevància les conferències rellevants per l'Autor x.

Consultes amb filtre

Per realitzar una consulta amb filtre, fer click al botó "Afegir filtre". Es pot configurar el filtre per tal que no apareguin els resultats que no tinguin una rellevància mínima o que superin una rellevància màxima. També es pot filtrar els resultats a partir del càlcul de la rellevància entre dues dades i una relació. Els resultats que tinguin una rellevància inferior a aquesta rellavància calculada no apareixeran. En aquest últim cas també se li aplica el filtre de mínima i màxima rellevància.

Consulta i gestió dels resultats

-Filtrar resultats

Es poden aplicar filtres al resultat d'una consulta.
Per quedar-se amb les x primeres dades rellevants del resultat, seleccionar la opció filtrar els "n" primers, introduint la "n" desitjada.
Per quedar-se amb les x últimes dades rellevants del resultat, seleccionar la opció filtrar els "n" últims, introduint la "n" desitjada.
Per quedar-se només amb les dades rellevants que estan associades a una etiqueta particular, seleccionar la opció filtrar per etiquetes i seleccionar la etiqueta desitjada.

-Modificar resultats

Es poden afegir, modificar o esborrar dades rellevants al resultat d'una consulta.
Per afegir una nova dada rellevant al resultat fer click al botó "Afegir dada" i introduir la informació necessària.
Per modificar o esborrar una dada rellevant fer click al botó "Modificar" o "Esborrar" de la dada rellevant desitjada. En el cas de modificar, introduir la informació pertinent.

-Exportar resultats

Per exportar un resultat fer click al botó "Exportar" a la part superior dreta de la finestra "Resultat".

-Recuperar resultats

Menú principal > Consulta > Recuperar consulta anterior

Per recuperar resultats de consultes anteriors seleccionar un dels resultats del selector "Recuperar consulta anterior".

Gestió de dades

Gestió de conjunts de dades

-Escollir conjunt de dades

El conjunt de dades sobre el que es realitzaran les accions que ordeni l'usuari es pot seleccionar en diferents finestres, a través d'un selector ubicat a la part alta de la finestra.
Es pot escollir conjunt de dades a les finestres Menú Principal, Consulta i Modificar Dades.

-Importar conjunt de dades

Menú principal > Gestió de dades > Gestor de conjunt de dades > Importar conjunt de dades

Per importar un conjunt de dades, introdueix el path del directori (directament o a través del botó “escollir directori amb dades d’importació”) i un nom per identificar el nou graf.
Per finalitzar el procés fer click al botó “importar graf”.

-Crear conjunt de dades

Menú principal > Gestió de dades > Gestor de conjunt de dades > Crear nou conjunt de dades

Per crear un nou conjunt de dades introduir el nom del nou graf i seguidament fer click al botó “OK”. Al finalitzar l’acció s’haurà crear un graf buit que s’identificarà pel nom introduit.

-Modificar conjunt de dades

Menú principal > Gestió de dades > Gestor de conjunt de dades > Modificar conjunt de dades

Afegir dades

Per afegir una dada és necessari indicar el tipus de la dada que es vol afegir (amb el selector “tipus”) i el nom de la nova dada. És opcional associar una etiqueta a la dada a través del selector “etiqueta”.
A més a més, es poden afegir relacions de la nova dada amb altres dades. Per cada nova relació que es vulgui afegir, s’ha de fer click al botó “Afegir Relació” i introduir el tipus i el nom de la dada relacionada.
A l’introduir el nom de la dada relacionada, se n’ha de seleccionar una entre les dades que apareixaran en una finestra amb totes les dades que coincideixen amb el nom introduit, permetent veure inormació addicional d’aquestes.

Modificar dades

Per modificar una dada és necessari indicar el tipus de la dada que es vol afegir (amb el selector “tipus”) i el nom de la dada a modificar.
A més a més, es poden afegir o esborrar relacions de la dada seleccionada. Per cada nova relació que es vulgui afegir, s’ha de fer click al botó “Afegir Relació” i introduir el tipus i el nom de la dada relacionada.
A l’introduir el nom de la dada relacionada, se n’ha de seleccionar una entre les dades que apareixeran en una finestra amb totes les dades que coincideixen amb el nom introduit, permetent veure inormació addicional d’aquestes.
Per esborrar una relació, fer click al botó “Esborrar” de la relació de la taula de relacions que es vol esborrar.

Esborrar dades

Seleccionar el tipus de dada a esborrar i seguidament seleccionar la dada que es vulgui esborrar i fer click al botó “Esborrar”. Es pot veure informació addicional de les dades fent click al botó “Informació addicional”.

-Esborrar conjunt de dades

Menú principal > Gestió de dades > Gestor de conjunt de dades > Modificar conjunt de dades > Esborrar conjunt

Fer click al botó “esborrar conjunt” i s’esborrarà el conjunt de dades que estigui seleccionat com a actual.

Gestió de relacions

Menú principal > Gestió de dades > Gestor de relacions

-Crear una nova relació

Per crear una relació fer click al botó “Afegir relació” i introduir el nom de la relació i la seva descripció.

-Esborrar una relació

Per esborrar una relació fer click al botó “Esborrar” de la relació que es vulgui esborrar.

Guardat de dades

Es pot fer un guardat general de les dades al Menú Principal fent click al botó “Guardar”.
Tot i així, quan es vol tancar el programa, abans de tancar es pregunta a l’usuari si vol fer un guardat general de totes les dades.

Descripció d'estructures de dades i algorismes

HeteSim

Per fer el càlcul del HeteSim, en tots el casos, primer descomposem el path i obtenim les matrius d'adjacència necessaries pel càlcul, posteriorment inserim les matrius intermèdies si el path té longitud senar. Les matrius resultants són guardades en un ArrayList per fer més fàcil la iteració sobre el contigut.

En el cas del càlcul de clausures, les matrius obtingudes de la descomposició es multipliquen des de la posició 0 fins a la meitat per obtenir la matriu left i des de la meitat fins al final per obtenir la matriu right. Després, normalitzem left per files i right per columnes i finalment obtenim la clausura multiplicant ambdues matrius i, només si el path tenia més de dos elements, normalitzem els seus elements aplicant la fòrmula del HeteSim. Es retorna una Matriu amb les rellevàncies entre els tipus de Node indicats pel path. Per tal de fer més eficients les consultes a la mateixa clausura aquesta es guarda a disc per no haver de tornar a calcular-la, existent la possibilitat de forçar que es recalculi. Aquestes clausures també mantenen un booleà que indica si estan actualitzades o no, tot i que no afecta en res al càlcul.

Per fer el càlcul de la relleància entre dos Nodes, s'agafa la fila corresponent al primer Node i es va fent la multiplicació amb les matrius següents (obtenint sempre una altra fila) fins la meitat, després agafem la columna de l'última matriu corresponent al segon Node i l'anem multiplicant amb les matrius anteriors (sempre obtenint una columna) fins la meitat. Posteriorment, es fa la multiplicació entre la fila i columnes resultants i es retorna el valor númeric resultant normalitzat amb la fòrmula del HeteSim.

Per fer el càlcul entre un Node i tots els Node d'un altre tipus, s'agafa la fila corresponent al Node i es va fent la multiplicació amb les matrius següents (obtenint sempre una altra fila) fins la meitat, després es multipliquen les matrius des de la meitat fins al final. Finalment es multiplica la fila per la matriu resultants i s'obté una llista de les rellevàncies entre el Node i tots els Node d'un altre tipus, ordenada decreixentment per rellevància. Finalment, segons si es vol tenir les rellevàncies amb el IDs del Node o bé amb els seus noms es retorna ArrayList<Entry<Double, Integer>> o un ArrayList<Entry<Double, String>>.

Resultat

La classe Resultat és una classe que conté informació sobre un resultat. Concretament conté informació sobre la dada, el path, el nom del graf i el filtre (threshold) a partir dels quals s'ha realitzat la consulta. A més, contindrà el conjunt de tuples (rellevància,dada) que s'han obtingut al fer la consulta.

Cal dir que la classe resultat només conté informació, sense mantenir totes les regles que s'haurien de complir per tal de que els resultats tinguin sentit, ja que l'usuari ha de poder modificar-lo al seu gust. L'única propietat que sempre es manté és que la llista serà decreixent segons la rellevància de cada tupla.
Per tant, un objecte de la classe por contenir informació incoherent. Per exemple, si es canvia el threshold, no es reajustaran els resultats al nou threshold, si es canvia la rellevància d'una tupla no es comprovarà que aquesta s'ajusti al threshold, i tampoc es comprovarà que els tipus dels nodes concordin amb el Path del resultat.

Per tal de facilitar la feina a la capa de presentació existeix la possibilitat de convertir els resultats de la consulta en tipus bàsics de Java, mitjançant una llista que conté tots els parells de rellevància i node, on node es representa com un parell amb el seu identificador i el seu nom. Aquestes conversions es guarden en un WeakHashMap que actua com a caché de dades temporal (s'eliminen entrades si falta memòria) per no repetir conversions mentre s'estableixen els resultats en la capa de presentació.

Matriu

La classe Matriu, és una implementació CSR d'una matriu esparsa. S'utilitzen 2 arrays i una llista per representar les dades no nul·les de la matriu que corresponen a un array dels valors no nuls (AA), els índexs a les seves columnes (JA) i els punters a les files (IA). La Matriu pot normalitzarse, en aquest cas, es calculen totes les normes una sola vegada i es guarden en un array de files normalitzades i un altre de columnes normalitzades per poder obtenir-los ràpidament en futurs accessos.

Tots els accesos en aquesta matriu es corresponen a accessos a valors no nuls. Si es fa un get(i, j) es consulta si hi ha un element no nul i es retorna 0 si no hi és o el valor en qüestió d'altra manera.

Quan s'afegeixen files o columnes a una nova matriu es crea simultáneament la matriu trasposada, per poder transposar directament la matriu sense recalcularla quan es necessiti.

Per fer la multiplicació A*B de matrius s'accedeixen a les files que contenen valor no nuls de A i es recorren els seus valors per columnes no nuls fent la multiplicació tradicional només amb els valors no nuls de la matriu B. Per fer-ho més eficient es desenrotlla el bucle per 4 i es fa inlining de la funció de binarySearch de Java per que no faci comprovacions d'excepcions a l'hora de buscar ordenadament en els índexos de columnes si existeix el valor a multiplicar.

Per fer les matrius intermèdies AE i EB donada una matriu AB va recorrent els valors no nuls de A i B i els possa en la posició escalonada de les matrius resultants AE i EB.

Graf

La classe Graf, conté els nodes (dades) i les seves adjacències (relacions). Els nodes es guarden en un HashMap<Integer, Node> identificats pel seu ID. Per facilitar les cerques de nodes per nom, guardem un HashMap<String, ArrayList<Node>> on la key es el nom i el valor es una llista de nodes, ja que els noms es poden repetir entre diversos nodes. Finalment, les adjacències es guarden en tres Matriu, una amb Paper-Autor, Paper-Conferència i Paper-Terme. Per guardar les adjacències, utilitzem els IDs dels nodes per establir quina fila/columna representen, els Paper sempre estan representats a les files i Autors, Papers i Conferències a les columnes. Així, per dos nodes R i S podem saber si tenen una adjacència si al mirar la posició definida per la fila R.id i la columna S.id observem un '1'.

Diagrames de classes

- Capa de presentació:http://imgur.com/nzxsvxa
- Capa de domini:http://imgur.com/PmYbmEx
- Capa de persistència:http://imgur.com/oEA3ulX

Repartiment de responsabilitats de classes compartides

Grup 1:

- Node
- Autor
- Paper
- Conferència
- Terme
- Graf
- ControladorNodes
- ControladorGraf

Grup 2:

- ControladorPersistència
- ControladorDominiPersistència
- HeteSim

Grup 3:

- Path
- ControladorPaths
- ControladorRelacions
- Matriu<T>

- # Especificació de classes compartides

Node

*classe **abstracta**, implements **Serializable***

Defineix un node del graf.

Atributs:

- *protected* **nom** : String
- *protected* **id** : Integer
- *protected* **label** : String

Mètodes:

- **Node**(): Constructor per defecte.
- **Node**(int id, String nom): Constructor amb id i nom.
- **Node**(int id, String nom, String label): Constructor amb id, nom i label.

- void **setLabel(**String label): Modificador de label.
- void **setNom**(String nom): Modificador de nom.
- void **setId**(int id): Modificador de id.

- String **getLabel**(): Consultor de label.
- String **getNom**(): Consultor de nom.
- int **getId**(): Consultor d'id.

- String **toString**(): Retorna un String que representa aquest objecte.

Autor

extends ***Node***

Subclasse de node que especifica que el node és un autor.

Atributs: -

Mètodes:

- **Autor**(): Constructor per defecte.
- **Autor**(int id, String nom): Constructor amb id i nom.
- **Autor**(int id, String nom, String label): Constructor amb id, nom i label.

Paper

extends ***Node***

Subclasse de node que especifica que el node és un paper (article).

Atributs: -

Mètodes:

- **Paper**(): Constructor per defecte.
- **Paper**(int id, String nom): Constructor amb id i nom.
- **Paper**(int id, String nom, String label): Constructor amb id, nom i label.

Conferencia

extends ***Node***

Subclasse de node que especifica que el node és una conferència.

Atributs: -

Mètodes:

- **Conferencia**(): Constructor per defecte.
- **Conferencia**(int id, String nom): Constructor amb id i nom.
- **Conferencia**(int id, String nom, String label): Constructor amb id, nom i label.

Terme

extends ***Node***

Subclasse de node que especifica que el node és un terme.

Atributs: -

Mètodes:

- **Terme**(): Constructor per defecte.
- **Terme**(int id, String nom): Constructor amb id i nom.

Matriu

implements ***Serializable***

Classe genèrica que representa una matriu amb elements de tipus T on totes les columnes tenen la mateixa mida.

Atributs:

- *private* **normalMat** : InternalMatrix
- *private* **transpMat** : InternalMatrix
- *private static* **aborted** : boolean
- *private static* **THRESHOLD** : double

Mètodes:

- **Matriu**(int files, int columnes, double elem): Constructor que crea una matriu amb un nombre determinat de files i columnes amb l'element elem en cada posició.
- *static* void **abort**() throws *InterruptedException*: Mètode static de la clase, que en esser cridat atura els calculs llençant una excepció que pot ser capturada desde presentacio.
- *static* void **reset**(): Mètode static de la clase, que en esser abortat un calcul cal cridar per tal de que les matrius puguin tornar a funcionar.

- doble **get**(int fila, int columna): Retorna l'element a la posició (fila, columna).

- void **set**(int fila, int columna, double elem): Posa elem a la posició (fila, columna).

- double **getNormaFila**(int fila): Retorna la norma per una fila.
- double **getNormaColumna**(int columna): Retorna la norma per una columna.

- int **afegirFila**(double elem): Afegeix una fila a la matriu amb elem en cada posició i retorna l'índex de la nova fila.
- int **afegirColumna**(double elem): Afegeix una columna a la matriu amb elem en cada posició i retorna l'índex de la nova columna.

- ArrayList<Double> **getFila**(int fila): Retorna una fila sencera tal com està la matriu.
- ArrayList<Double> **getColumna**(int columna): Retorna una fila sencera tal com està la matriu.
- ArrayList<Double> **getFilaNormalitzada**(int fila): Retorna una fila normalitzada.
- ArrayList<Double> **getColumnaNormalitzada**(int columna): Retorna una columna normalitzada.

- boolean **esborrarFila**(int fila): Esborra una fila de la matriu i retorna si ha sigut possible.
- boolean **esborrarColumna**(int columna): Esborra una columna de la matriu i retorna si ha sigut possible.

- int **getFiles**(): Consulta el número de files de la matriu.
- int **getColumnes**(): Consulta el número de columnes de la matriu.

- Matriu **multiplicar**(Matriu m) throws *IllegalArgumentException*: Multiplica el paràmetre implícit amb m i ho retorna. Llença una excepció en cas de que m i el paràmetre implícit no siguin compatibles per la multiplicació (this.getColumnes() == m.getFiles()).
- Matriu **transposada**(): Retorna la matriu transposada d'aquesta. El paràmetre implícit no queda modificat.

- void **transposar**(): Transposa el paràmetre implícit.
- Matriu **normalitzadaPerFiles**(): Retorna la matriu normalitzada del paràmetre implícit per files. El paràmetre implícit no queda modificat.
- Matriu **normalitzadaPerColumnes**(): Retorna la matriu normalitzada del paràmetre implícit per columnes. El paràmetre implícit no queda modificat.
- void **normalitzaPerFiles**(): Normalitza el paràmetre implícit per files.
- void **normalitzaPerColumnes**(): Normalitza el paràmetre implícit per columnes.
- ArrayList<Matriu> **intermedia**(): Retorna una llista amb les matrius intermèdies del parametre implicit. Per exemple, donada una matriu AP retorna les matrius AE i EP, on E és la matriu intermèdia.
- String **toString**(): Retorna un String que representa aquest objecte.

Graf

*implements **Serializable***

Representa la base de dades del problema que ens ocupa.

Atributs:

- *private* **adjPapersAutors** : Matriu
- *private* **adjPapersTermes** : Matriu
- *private* **adjPapersConferencies** : Matriu
- *private* **idToAutors** : HashMap<Integer, Autor>
- *private* **idToPapers** : HashMap<Integer, Paper>
- *private* **idToConferencies** : HashMap<Integer, Conferencia>
- *private* **idToTermes** : HashMap<Integer, Terme>
- *private* **nomToAutors** : HashMap<String, ArrayList<Autor>>
- *private* **nomToPapers** : HashMap<String, ArrayList<Paper>>
- *private* **nomToConferencies** : HashMap<String, ArrayList<Conferencia>>
- *private* **nomToTermes** : HashMap<String,ArrayList<Terme>>

Mètodes:

- **Graf**(): Constructor per defecte del graf.

- int **afegeix**(Autor autor): Afegeix un autor al graf sense cap relació i retorna el nou identificador intern.
- int **afegeix**(Paper paper): Afegeix un paper al graf sense cap relació i retorna el nou identificador intern.
- int **afegeix**(Conferencia conferencia): Afegeix una conferència al graf sense cap relació i retorna el nou identificador intern.
- int **afegeix**(Terme terme): Afegeix una Terme al graf sense cap relació i retorna el nou identificador intern.

- boolean **elimina**(Autor autor): Elimina un autor del graf i retorna si ha sigut possible.
- boolean **elimina**(Paper paper): Elimina un paper del graf i retorna si ha sigut possible.
- boolean **elimina**(Conferencia conferencia): Elimina una conferència del graf i retorna si ha sigut possible.
- boolean **elimina**(Terme terme): Elimina una Terme del graf i retorna si ha sigut possible.

- int **consultaMidaAutor**(): Consulta el nombre d'autors al graf.
- int **consultaMidaPaper**(): Consulta el nombre de papers al graf.
- int **consultaMidaConferencia**(): Consulta el nombre de conferències al graf.
- int **consultaMidaTerme**(): Consulta el nombre de termes / temàtiques al graf.

- boolean **afegirAdjacencia**(Paper paper, Autor autor): Afegeix la relació entre el paper i l'autor. Retorna si ha sigut possible.
- boolean **afegirAdjacencia**(Paper paper, Conferencia conferencia): Afegeix la relació entre el paper i la conferència. Retorna si ha sigut possible.

- boolean **afegirAdjacencia**(Paper paper, Terme tem): Afegeix la relació entre el paper i la Terme. Retorna si ha sigut possible.
- boolean **setAdjacencia**(Paper paper, Conferencia conferencia): Modifica la relació que tenia el paper amb l'anterior conferència i la canvia per la nova. Retorna si ha sigut possible.

- boolean **eliminarAdjacencia**(Paper paper, Autor autor): Elimina la relació entre el paper i l'autor.
- boolean **eliminarAdjacencia**(Paper paper, Conferencia conferencia): Elimina la relació entre el paper i la conferència. Retorna si ha sigut possible.
- boolean **eliminarAdjacencia**(Paper paper, Terme tem): Elimina la relació entre el paper i la Terme. Retorna si ha sigut possible.

- boolean **contains**(Autor autor): Consulta si l'autor és al graf.
- boolean **contains**(Paper paper): Consulta si el paper és al graf.
- boolean **contains**(Conferencia conferencia): Consulta si la conferència és al graf.
- boolean **contains**(Terme terme): Consulta si el terme és al graf.

- Autor **consultarAutor**(int idAutor): Retorna l'autor amb l'id corresponent o null si no existeix.
- List<Autor> **consultarAutor**(String nom): Retorna tots els autors amb el nom corresponent.
- Paper **consultarPaper**(int idPaper): Retorna el paper amb l'id corresponent o null si no existeix.
- List<Paper> **consultarPaper**(String nom): Retorna tots els papers amb el nom corresponent.
- Conferencia **consultarConferencia**(int idConferencia): Retorna la conferència amb l'id corresponent o null si no existeix.

- List<Conferencia> **consultarConferencia**(String nom): Retorna totes les conferències amb el nom corresponent.
- Terme **consultarTerme**(int idTerme): Retorna la Terme amb l'id corresponent o null si no existeix.
- List<Terme> **consultarTerme**(String nom): Retorna totes les temàtiques amb el nom corresponent.

- ArrayList<Autor> **consultarAutors()**: retorna tots els autors.
- ArrayList<Paper> **consultarPapers**(): retorna tots els papers.
- ArrayList<Conferencia> **consultarConferencies**(): retorna totes les conferències.
- ArrayList<Terme> **consultarTermes**(): retorna tots els termes.

- Matriu **consultarMatriuPaperAutor**(): Retorna la matriu d'adjacència de papers i autors.
- Matriu **consultarMatriuPaperConferencia**(): Retorna la matriu d'adjacència de papers i conferències.
- Matriu **consultarMatriuPaperTerme**(): Retorna la matriu d'adjacència de papers i temàtiques / termes.

Path

*implements **Serializable***

Classe que representa un path / relació entre dos tipus de nodes.

Atributs:

- *private* **path** : String
- *private* **definicio** : String

Mètodes:

- **Path**(String path): Constructor amb path.
- **Path**(String path, String definicio): Constructor amb path i def.

- String **getPath**(): Retorna el path.
- String **getDefinicio**(): Retorna la definició o descripció.

- void **setDefinicio**(String definicio): Modifica la definició amb la nova.
- String **toString**(): Retorna un String que representa aquest objecte.

ControladorPersistencia

Classe ***abstracta*** (només per evitar que s'instancii ja que els seus mètodes són estàtics)

Permet intercanviar dades amb la memòria secundària.

Atributs: -

Mètodes:

- *static* ArrayList<String> **llegirFitxer**(String filesystem_path) *throws IOException*: Llegeix el fitxer corresponent al path donat i retorna totes les línies del fitxer. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer.

- *static* void **esborrarFitxer**(String filesystem_path) throws *IllegalArgumentException*: Esborra el fitxer indicat, si existeix. Si es tracta d'un directori, el borra recursivament. Llençarà una *IllegalArgumentException* si filesystem_path es nul.

- *static* void **copiar**(String pathSrc, String pathDest) throws *IllegalArgumentException, IOException*: copia un fitxer o directori a un altre. Llençarà una *IllegalArgumentException* si algún dels paths es nul i una *IOException* si no es pot llegir o escriure a algún dels fitxers.

- *static* <T extends Serializable> void **guardar**(String filesystem_path, T dada) throws *IOException*: Guarda un tipus de dada serialtzable a un fitxer. Llençarà una *IOException* si no es pot escriure en el fitxer.

- *static* <T extends Serializable> T **carregar**(String filesystem_path) throws *IOException*: Llegeix una dada serialitzada des d'un fitxer. Llençarà una *IOException* si no es pot llegir del fitxer o no conté una dada serialitzada.

- *static* void **guardarGraf**(String filesystem_path, Graf graf) *throws IOException*: Guarda el graf serialitzable a un fitxer corresponent al path donat, sobreescrivint-lo si ja existeix. Llençarà una excepció si no es pot escriure en el fitxer.

- *static* Graf **carregarGraf**(String filesystem_path) *throws IOException*: Llegeix un graf serialitzat del fitxer corresponent al path donat Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

- *static* void **guardarPaths**(String filesystem_path, Collection<Path> paths) *throws IOException*: Guarda una colecció de paths serialitzables a un fitxer corresponent al path donat, sobrescrivint-lo si ja existeix. Llençarà una excepció si no es pot escriure en el fitxer.

- *static* Collection<Path> **carregarPaths**(String filesystem_path) *throws IOException*: Llegeix una colecció de paths serialitzats que representa una colecció de paths del fitxer corresponent al path donat. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

- *static* void **guardarClausura**(String dir, String id, Matriu clausura, boolean updated) throws *IOException*: Guarda una clausura amb el seu identificador al directori dir. Lleçarà una *IOException* si no es pot crear o escriure en el fitxer.

- *static* void **actualitzaClausura**(String dir, String id, boolean updated) throws *IOException*: Defineix com a actualitzada o desactualitzada la clausura amb identificador id al directori dir. Llençarà una *IOException* si no es pot actualitzar el fitxer.

- *static* void **isClausuraActualitzada**(String dir, String id) throws *FileNotFoundException*: Retorna si la clausura amb identificador id al directori dir, està actualitzada. Llençarà una *FileNotFoundException* si el directori és nul o el directori no existeix.

- *static* void **esborrarClausura**(String dir, String id) throws *IOException*: esborra la clausura amb identificador id al directori dir. Llençarà una *IOException* si no es pot esborrar el fitxer.

- *static* boolean **existsClausura**(String dir, String id) throws *IOException*: retorna si existeix la clausura amb identificador id al directori dir. Llençarà una *FileNotFoundException* si dir es nul o no és un directori existent.

- *static* HashMap<String, Boolean> **carregarClausures**(String dir) throws *IOException*: retorna els noms de totes les clausures al directori dir i si estan actualitzades. Llençarà una *IOException* si no es pot llegir algún fitxer o no hi ha un objecte amb el format correcte.

- *static* Matriu **carregarClausura**(String dir, String id) throws *IOException*: llegeix la clausura serialitzada amb identificador id des del directori dir. Llençarà una *IOException* si no es pot llegir el fitxer, no hi ha un objecte serialitzat o no es troba la classe serialitzada.

ControladorPaths

implements Serializable

Permet mantenir un conjunt de paths.

Atributs:

- *private* **paths** : Map<String, Path>

Mètodes:

- **ControladorPaths**(ControladorGraf controladorGraf)

- boolean **afegir**(String path, String definicio): Afegeix un nou path i retorna si ha sigut possible (encara no hi era).

- boolean **afegir**(String path): Afegeix un nou path sense definició i retorna si ha sigut possible (encara no hi era).

- boolean **modificarDefinicio**(String path, String definicio): Modifica la definció d'un path i retorna si ha sigut possible (si existía).

- boolean **esborrar**(String path): Esborra un path i retorna si ha sigut possible (si existía).

- List<String> **consultarPaths**(): Retorna una llista amb tots els paths.
- String **consultarDefinicio**(String path): Retorna la definició d'un path.

- void **guardarPaths**(String filesystem_path) *throws IOException:* Guarda tots els paths en el fitxer donat. Si el fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en el fitxer.

- void **carregarPaths**(String filesystem_path) *throws IOException:* Carrega tots els paths des d'un fitxer. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer.

HeteSim

implements ***Serializable***

Controlador de l'algorisme.

Atributs:

- *protected* **graf** : Graf
- *private* **directoriClausures** : String
- *public static final* **DEFAULT_DIRECTORI_CLAUSURES** : String

Mètodes:

- HeteSim(Graf graf, String directoriClausures)
- HeteSim(Graf graf)

- Graf getGraf(): Retorna el graf que s'utilitza per calcular l'algorisme.

- Matriu clausura(String path) throws *IllegalArgumentException, IOException, InterruptedException*: Retorna la matriu de clausura d'un path donat. La matriu de clausura és la matriu on cada element (i, j) és la rellevància de l'element representat a la fila i amb l'element representat a la fila j. Llençarà una excepció en cas de que path sigui null.

- Matriu clausura(String path, boolean ignorarClausura) throws *IllegalArgumentException, IOException, InterruptedException*: Retorna la matriu de clausura d'un path donat. La matriu de clausura és la matriu on cada element (i, j) és la rellevància de l'element representat a la fila i amb l'element representat a la fila j. Llençarà una excepció en cas de que path sigui null. Si ignorarClausura llavors qualsevol clausura existent per aquest path serà ignorada i es tornarà a calcular.

- double heteSim(Node a, Node b, String path) throws *IllegalArgumentException, IOException, InterruptedException*: Retorna la rellevància entre dos nodes a i b donat un path. Llençarà una excepció en cas de que path sigui null o bé a no sigui del tipus indicat pel primer tipus de node del path o bé b no sigui del tipus indicat pel últim tipus de node del path que es passa com a paràmetre.

- double heteSim(Node a, Node b, String path, boolean ignorarClausura) throws *IllegalArgumentException, IOException, InterruptedException*: Retorna la rellevància entre dos nodes a i b donat un path. Llençarà una excepció en cas de que path sigui null o bé a no sigui del tipus indicat pel primer tipus de node del path o bé b no sigui del tipus indicat pel últim tipus de node del path que es passa com a paràmetre. Si ignorarClausura llavors qualsevol clausura

existent serà ignorada i es calcularà la rellevància, en cas contrari el valor de la rellevància s'extreurà de l'última clausura calculada per aquest path.

- ArrayList<Entry<Double, Integer>> heteSimAmbIdentificadors(Node n, String path) throws *IllegalArgumentException, IOException, InterruptedException*: Retorna una llista ordenada per rellevàncies on cada element és la rellevància que té n amb un altre node representat pel seu identificador amb un path donat. Llençarà una excepció en cas de que path sigui null o bé que n no sigui del tipus indicat pel primer tipus de node del path.

- ArrayList<Entry<Double, Integer>> heteSimAmbIdentificadors(Node n, String path,boolean ignorarClausura) throws *IllegalArgumentException, IOException, InterruptedException*: Retorna una llista ordenada per rellevàncies on cada element és la rellevància que té n amb un altre node representat pel seu identificador amb un path donat. Llençarà una excepció en cas de que path sigui null o bé que n no sigui del tipus indicat pel primer tipus de node del path. Si ignorarClausura llavors qualsevol clausura existent serà ignorada i es calcularan les rellevàncies, en cas contrari el valors de la rellevàncies s'extreurà de l'última clausura calculada per aquest path.

- ArrayList<Entry<Double, String>> heteSimAmbNoms(Node n, String path) throws *IllegalArgumentException, IOException, InterruptedException*: Retorna una llista ordenada per rellevàncies on cada element és la rellevància que té n amb un altre node representat pel seu nom amb un path donat. Llençarà una excepció en cas de que path sigui null o bé que n no sigui del tipus indicat pel primer tipus de node del path.

- ArrayList<Entry<Double, String>> heteSimAmbNoms(Node n, String path, boolean ignorarClausura) throws *IllegalArgumentException, IOException, InterruptedException*: Retorna una llista ordenada per rellevàncies on cada element és la rellevància que té n amb un altre node representat pel seu nom amb un path donat. Llençarà una excepció en cas de que path sigui null o bé que n no sigui del tipus indicat pel primer tipus de node del path. Si ignorarClausura llavors qualsevol clausura existent serà ignorada i es

calcularan les rellevàncies, en cas contrari el valors de la rellevàncies s'extreurà de l'última clausura calculada per aquest path.

- void eliminarClausura(String path) throws *IOException*: esborra una clausura de l'Hetesim, si existeix. Llençarà una excepció en cas que no es pugui esborrar la clausura.
- HashMap<String, Boolean> getClausures() throws *IOException*: retorna totes les clausures disponibles per aquest HeteSim amb el seu path com a identificador i si està actualitzada. Llençarà una excepció si no es poden llegir les clausures.

- boolean existsClausura(String path): retorna si existeix la clausura identificada per path.

- boolean isUpdated(String path): si la clausura existeix, retorna si està actualitzada.

- void setUpdated(String path, boolean updated) throws *IOException*: defineix si la clausura identificada per path està actualitzada o no en funció del valor de updated. Llençarà una excepció si no es pot llegir o escriure al fitxer de clausures.

- void validarClausures() throws *IOException*: defineix totes les clausures de l'Hetesim actual com a actualitzades. Llençarà una excepció si no es pot llegir o escriure al fitxer de clausures.

- void invalidarClausures() throws *IOException*: defineix totes les clausures de l'Hetesim actual com a desactualitzades. Llençarà una excepció si no es pot llegir o escriure al fitxer de clausures.

- void setUpdatedAll(String regex, boolean updated) throws *IOException, PatternSyntaxException*: actualitza o desactualitza totes les clausures que compleixen l'expresió regular regex, en funció del valor d'updated.

- void setUpdatedAll(boolean updated) throws *IOException*: actualitza o desactualitza totes les clausures en funció del valor d'updated.

- boolean esGuardenClausures(): retorna si es guarden les clausures d'aquest HeteSim a disc.

ControladorNodes

Controlador amb funcions sobre un node.

Atributs:

- *private* **controladorGraf** : ControladorGraf

Mètodes:

- **ControladorNodes**(ControladorGraf controladorGraf)

- int **afegirAutor**(String nom): Afegeix un autor sense cap etiqueta al graf actual i retorna el seu identificador intern.
- int **afegirAutor**(String nom, String label): Afegeix un autor al graf actual i retorna el seu identificador intern.
- int **afegirLabelAutor**(String label, int idAutor) throws *IllegalArgumentException*: Afegeix una etiqueta a un autor del graf i retorna el seu identificador intern. Llençarà una excepció en cas de que autor no existeixi en el graf actual.

- int **afegirConferencia**(String nom): Afegeix una conferència al graf actual sense cap etiqueta i retorna el seu identificador intern.
- int **afegirConferencia**(String nom, String label): Afegeix una conferència al graf actual i retorna el seu identificador intern.

- int **afegirLabelConferencia**(String label, int idConferencia) throws *IllegalArgumentException*: Afegeix una etiqueta a una conferència del graf i retorna el seu identificador intern. Llençarà una excepció en cas de que conferencia no existeixi en el graf actual.

- int **afegirPaper**(String nom): Afegeix un paper al graf actual sense cap etiqueta i retorna el seu identificador intern.
- int **afegirPaper**(String nom, String label): Afegeix un paper al graf actual i retorna el seu identificador intern.
- int **afegirLabelPaper**(String label, int idPaper) throws *IllegalArgumentException*: Afegeix una etiqueta a un paper del graf i retorna el seu identificador intern. Llençarà una excepció en cas de que paper no existeixi en el graf actual.

- int **afegirTerme**(String nom): Afegeix un terme al graf actual i retorna el seu identificador intern.

- boolean **modificarAutor**(String nouNom, int idAutor): Modifica el nom d'un autor. Retorna si ha sigut possible (si existeix).
- boolean **modificarPaper**(String nouNom, int idPaper): Modifica el nom d'un paper. Retorna si ha sigut possible (si existeix).
- boolean **modificarConferencia**(String nouNom, int idConferencia): Modifica el nom d'una conferència. Retorna si ha sigut possible (si existeix).
- boolean **modificarTerme**(String nouNom, int idTerme): Modifica el nom d'un terme. Retorna si ha sigut possible (si existeix).

- boolean **eliminarAutor**(int idAutor): Elimina un autor. Retorna si ha sigut possible (si existía).

- boolean **eliminarPaper**(int idPaper): Elimina un paper. Retorna si ha sigut possible (si existía).
- boolean **eliminarConferencia**(int idConferencia): Elimina una conferència. Retorna si ha sigut possible (si existía).
- boolean **eliminarTerme**(int idTerme): Elimina un terme. Retorna si ha sigut possible (si existía).

- String **consultarNomAutor**(int idAutor): Consulta el nom d'un autor.
- String **consultarNomPaper**(int idPaper): Consulta el nom d'un paper.
- String **consultarNomConferencia**(int idConferencia): Consulta el nom d'una conferència.
- String **consultarNomTerme**(int idTerme): Consulta el nom d'un terme.

- String **consultarLabelAutor**(int idAutor): Consulta l'etiqueta d'un autor.
- String **consultarLabelPaper**(int idPaper): Consulta l'etiqueta d'un paper.
- String **consultarLabelConferencia**(int idConferencia): Consulta l'etiqueta d'una conferència.

ControladorRelacions

Controlador de les relacions en un graf.

Atributs:

- *private* **controladorGraf** : ControladorGraf

Mètodes:

- **ControladorRelacions**(ControladorGraf controladorGraf)

- boolean **existeixRelacioPaperAutor**(int idPaper, int idAutor): Retorna cert si existeix la relació entre el paper i l'autor indicats.

- boolean **existeixRelacioPaperTerme**(int idPaper, int idTerme): Retorna cert si existeix la relació entre el paper i el terme indicats.
- boolean **existeixRelacioPaperConferencia**(int idPaper, int idConferencia): Retorna cert si existeix la relació entre el paper i la conferència indicats.

- boolean **afegirAdjacenciaPaperAutor**(int idPaper, int idAutor): Afegeix una adjacència al graf actual entre un paper i un autor. Retorna si ha sigut possible (si ambdós existeixen).
- boolean **afegirAdjacenciaPaperTerme**(int idPaper, int idTerme): Afegeix una adjacència al graf actual entre un paper i un terme. Retorna si ha sigut possible (si ambdós existeixen).

- boolean **setAdjacenciaPaperConferencia**(int idPaper, int idConferencia): Afegeix una adjacència al graf actual entre un paper i una conferència. En cas de que hi hagués prèviament una altra relació entre aquests dos nodes aquesta és eliminada ja que un paper només pot estar relacionat amb una conferència. Retorna si ha sigut possible (si ambdós existeixen).

- boolean **eliminarAdjacenciaPaperAutor**(int idPaper, int idAutor): Elimina la adjacència entre un paper i un autor. Retorna si ha sigut possible (si ambdós existeixen).
- boolean **eliminarAdjacenciaPaperTerme**(int idPaper, int idTerme): Elimina la adjacència entre un paper i un terme. Retorna si ha sigut possible (si ambdós existeixen).

- List<String> **consultarRelacionsConferencia**(int idConferencia): Consulta totes les relacions d'una conferència. La llista retornada conté els noms de tots els papers relacionats.

- List<String> **consultarRelacionsAutor**(int idAutor): Consulta totes les relacions d'un autor. La llista retornada conté els noms de tots els papers relacionats.
- List<String> **consultarRelacionsTerme**(int idTerme): Consulta totes les relacions d'un terme. La llista retornada conté els noms de tots els papers relacionats.
- List<String> **consultarRelacionsPaperAmbAutor**(int idPaper): Consulta totes les relacions d'un paper. La llista retornada conté els noms de tots els autors relacionats.
- List<String> **consultarRelacionsPaperAmbTerme**(int idPaper): Consulta totes les relacions d'un paper. La llista retornada conté els noms de tots els termes relacionats.
- List<String> **consultarRelacionsPaperAmbConferencia**(int idPaper): Consulta totes les relacions d'un paper. La llista retornada conté els noms de totes les conferències relacionades.

ControladorGraf

Controlador per mantenir l'estat del graf.

Atributs:

- *protected* **graf** : Graf
- protected **hetesim**: HeteSim

Mètodes:

- **ControladorGraf**()

- Graf **getGraf**(): Obté el graf actual.
- HeteSim **getHeteSim**(): retorna una referència a la instància de la classe HeteSim del paràmetre implícit.

- void **importar**(String directori) throws *FileNotFoundException, IOException:* Importa i crea el graf mitjançant els fitxers del directori que es passen com a paràmetre (autors, papers, conferències, labels i relacions). Llençarà una excepció de tipus *FileNotFoundException* en cas de que algun fitxer no es trobi o altre excepció de tipus *IOException* en cas de que algun fitxer no tingui el format correcte o bé no sigui possible llegir-lo.

- void **carregar**(String directori) throws *FileNotFoundException, IOException:* Carrega el graf del fitxer corresponent al path donat. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer.

- void **guardar**(String directori) throws *IOException:* Guarda el graf actual en el fitxer donat. Si el fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en el fitxer.

- List<Integer> **consultarPaper**(String nom): Retorna una llista amb els identificadors de tots els papers amb el nom del paràmetre.
- List<Integer> **consultarConferencia**(String nom): Retorna una llista amb els identificadors de totes les conferències amb el nom del paràmetre.
- List<Integer> **consultarTerme**(String nom): Retorna una llista amb els identificadors de tots els termes amb el nom del paràmetre.
- List<Integer> **consultarAutor**(String nom): Retorna una llista amb els identificadors de tots els autors amb el nom del paràmetre.

- TreeMap<Integer,String> **consultarPapers**(): Retorna un TreeMap que com a clau té el identificadors dels Papers i com a valor el nom d'aquests

- TreeMap<Integer,String> **consultarAutors**(): Retorna un TreeMap que com a clau té el identificadors dels Autors i com a valor el nom d'aquests
- TreeMap<Integer,String> **consultarConferencies**(): Retorna un TreeMap que com a clau té el identificadors de les conferències i com a valor el nom d'aquestes
- TreeMap<Integer,String> **consultarTermes**(): Retorna un TreeMap que com a clau té el identificadors dels Termes i com a valor el nom d'aquests

- int **consultarMidaAutors**(): Consulta quants autors hi ha en el graf actual.
- int **consultarMidaPapers**(): Consulta quants papers hi ha en el graf actual.
- int **consultarMidaTermes**(): Consulta quants termes / temàtiques hi ha en el graf actual.
- int **consultarMidaConferencies**(): Consulta quantes conferències hi ha en el graf actual.

ControladorDominiPersistencia

Implements ***Serializable***

Atributs:

- *public static final* **DEFAULT_FILEPATH_GRAF** : String = "graf.dat"
- *public static final* **DEFAULT_FILEPATH_PATHS** : String = "paths.dat"
- *protected* **controladorGraf** : ControladorGraf
- *protected* **controladorPaths** : ControladorPaths

Mètodes:

- **ControladorDominiPersistencia**(ControladorGraf controladorGraf, ControladorPaths controladorPaths, HeteSim controladorHeteSim)

- **ControladorDominiPersistencia**(ControladorGraf controladorGraf, ControladorPaths controladorPaths)

- void **guardarDades**() *throws IOException:* Guarda totes les dades que es poden (fa una crida als mètodes de guardar graf i paths) als fitxers corresponents als paths per defecte. Si algun fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en els fitxers.

- void **guardarDades**(String fitxerGraf, String fitxerPaths) *throws IOException:* Guarda totes les dades (fa una crida als mètodes de carregar graf i paths) als fitxers corresponents als paths donats per paràmetre. Si algun fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en els fitxers.

- void **carregarDades**() *throws IOException:* Carrega totes les dades (fa una crida als mètodes de guardar graf i paths) pels fitxers per defecte existents. Llença una excepció en cas de no poder llegir els fitxers.

- void **carregarDades**(String fitxerGraf, String fitxerPaths) *throws IOException:* Carrega totes les dades (fa una crida als mètodes de guardar graf i paths) pels fitxers corresponents als paths donats per paràmetre. Llençarà una *FileNotFoundException* en cas de que no es trobi algun fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

Pair<K extends Comparable<K> & Serializable, V>

implements ***Entry<K, V>, Comparable<Pair<K, V>>, Serializable***

Atributs:

- *private* **key** : K
- *private* **value** : V

Mètodes:

- **Pair**(K key, V value)

- K **getKey**(): Retorna la clau del pair.
- V **getValue**(): Retorna el valor del pair.

- void **setKey**(K key): Canvia la clau del pair.
- V **setValue**(V value): Canvia el valor del pair i retorna el valor antic.

- int **compareTo**(Pair<K, V> p): Retorna un nombre positiu, 0 o un nombre negatiu segons si this és menor, igual o major a p, respectivament. La comparació es fa entre les claus dels dos pairs.

- String **toString**(): Retorna un String que representa aquest objecte.

Diferències amb la 1ª entrega

Casos d'ús afegits:

- ***Modificar dades rellevants d'un resultat***: Canvia la dada o la rellevància d'una d'una dada rellevant d'un resultat.

- ***Afegir dades rellevants a un resultat***: S'afegeix una dada rellevant (amb la seva corresponent rellevància) al resultats.

Casos d'ús no implementats:

- ***Canviar paràmetres dels resultats***: No es poden recuperar els paràmetres dels resultats d'una consulta prèvia perquè es pot donar el cas de que les dades emprades per fer la consulta siguin esborrades del conjunt de dades i llavors no es puguin tornar a carregar com a paràmetres. El que sí es pot fer es obtenir els resultats i paràmetres d'una consulta anterior, però només per visualitzar-ho i aplicar-hi filtres.

Repartiment de responsabilitats

Arnau Badia Sampera

- *ControladorDominiPersistència*
- ControladorDominiPersistènciaPropi
- ControladorMultigraf
- Resultat
- ControladorPresentació
- MenuPrincipal
- InformacioAddicional
- GestorConjuntDades
- Importar

Carla Claverol González

- ControladorConsultes
- *Pair<K, V>*
- Threshold
- GestorDades
- GestorRelacions
- AfegirRelacio
- ModificarDescripcio
- Resultats
- AfegirResultat
- ModificarResultat

Carlos Lázaro Costa

- *HeteSim*
- *ControladorPersistència*
- ControladorExportació
- TaskConsole
- SelectorConjunts
- ModificarConjuntDades
- Message
- ErrorMessage
- Consulta
- BounceProgressBarTaskFrame
- BounceProgressBarTask
- TipusDada

Guillem Castro Olivares

- *HeteSim*
- ControladorPersistènciaPropi
- AfegirDada
- EsborrarDada
- ModificarDada
- SeleccionarDada
- SeleccionarConferencia