Títol: **BibTEX Bibliography Index Maker**

Volum: **1/1**
Alumne: **Ramon Xuriguera Albareda**

Director/Ponent: **Marta Arias**
Departament: **LSI**
Data: **Primavera 2010**

DADES DEL PROJECTE

Títol del Projecte:

Nom de l'estudiant: Ramon Xuriguera Albareda

Titulació: Enginyeria Informàtica

Crèdits: 37,5

Director/Ponent: Marta Arias

Departament: LSI

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President:

Vocal:

Secretari:

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

Índex

Capítol 1

Introducció

Una mica de text per aquí

El format PDF ha esdevingut un estàndard per la divulgació de publicacions on-line.

Actualment existeixen serveis com ara *Google Scholar*, *Microsoft Academic Search* o *CiteSeer* que es dediquen a recol·lectar informació sobre articles i que comptabilitzen les referències entre diferents publicacions. En el cas de *CiteSeer*, al ser un projecte lliure, sabem que funciona analitzant les diferents parts dels articles, com ara les cites, però que també té problemes per obtenir els camps de la capçalera, que és el que ens interessa [GBL98].

Per una altra banda, també existeixen nombroses aplicacions dedicades al maneig de referències com *JabRef*[1] o *Mendeley*[2]. Entre algunes de les funcionalitats addicionals que ofereixen, hi ha la possibilitat de cercar en bases de dades d'articles i utilitzar les meta-dades dels fitxers per tal de trobar informació com ara el títol o l'autor. A banda d'això, no n'hem trobat cap que aprofiti el contingut dels documents per generar la referència.
El nostre sistema mira d'omplir el buit que queda entre aquestes dos tipus de programari que *BibTEX Bibliography Index Maker* és una eina d'ajuda a la creació d'índexs bibliogràfics pensada com un complement a aplicacions de maneig de referències ja existents com poden ser .

La principal funcionalitat que ofereix consisteix en escanejar un directori que conté articles científics en PDF i generar un índex bibliogràfic en BibTEX amb les referències d'aquests fitxers. Aquest índex es pot importar des de les aplicacions esmentades o bé pot ser referenciat directament des d'un nou document TEX.
Els requisits han estat definits per nosaltres mateixos.

1.1 Sobre aquest document

La memòria està dividida en els capítols següents:

- Capítol 2: Descripció formal del projecte així com un breu repàs sobre el disseny disseny i implementació.

[1] http://jabref.sourceforge.net
[2] http://www.mendeley.com

- Capítol 3: Parla de les tècniques que s'utilitzen per poder aconseguir pàgines que continguin informació sobre els documents pels quals es volen obtenir referències.
- Capítol 4: Tracta de com ho fem per extreure la informació sobre els articles de les pàgines d'Internet que hem obtingut al tercer capítol.
- Capítol 5: Està dedicat a les tècniques per generar, de forma automàtica, les regles d'extracció de les referències.
- Capítol 6: Plantejament de les proves per cadascuna de les parts més importants del sistema i anàlisi dels resultats obtinguts.

Respecte al contingut de les diferents seccions, hem provat de centrar-nos, sobretot, en les decisions que hem hagut de prendre al llarg de la realització del projecte; indicant-ne algunes de les opcions plantejades en un principi, els problemes que aquestes presenten i la descripció de la solució escollida finalment. S'han intentat ometre aspectes més tècnics sobre com s'ha implementat el sistema, així com descripcions de les tecnologies utilitzades.

S'assumeix que es tenen nocions bàsiques de l'estructura dels documents HTML i d'expressions regulars, dos temes que no es tractaran. Pel que fa a les expressions regulars, s'utilitzen les operacions que ofereix el mòdul `re` de *Python*. Es pot trobar més informació a [Pytc].

Capítol 2

Definició del Projecte

2.1 Requeriments

A continuació es llisten totes les funcionalitats que l'aplicació haurà d'oferir a l'usuari. Cal comentar que aquests requeriments els hem establert nosaltres mateixos a segons el que ens ha semblat més necessari.

- Extracció de la referència bibliogràfica corresponent a un o més articles que es troben en fitxers PDF dins d'algun directori.
- Possibilitat d'exportar les referències extretes en el format BibTeX i desar-les a un fitxer `.bib`
- Generació automàtica de regles d'extracció a partir d'exemples.
- Importació de referències en un fitxer `.bib` per tal de poder-los fer servir d'exemples.
- Totes les operacions CRUD[1] per a la gestió de referències i regles d'extracció.

2.2 L'extracció de referències

Aquest és l'objectiu principal de l'aplicació i les limitacions que hem trobat per aconseguir-lo són les que han guiat el disseny de la resta del sistema. A continuació es pretén mostrar, a grans trets, el raonament seguit a l'hora de definir una estratègia amb certes garanties d'èxit. En un principi, la idea que vam plantejar per resoldre el problema de l'extracció de referències d'un document PDF era intentar agafar la informació de la referència bibliogràfica directament del fitxer, seguint uns passos semblants als de la figura següent.

[1] *Create*, *read*, *update* i *delete*.

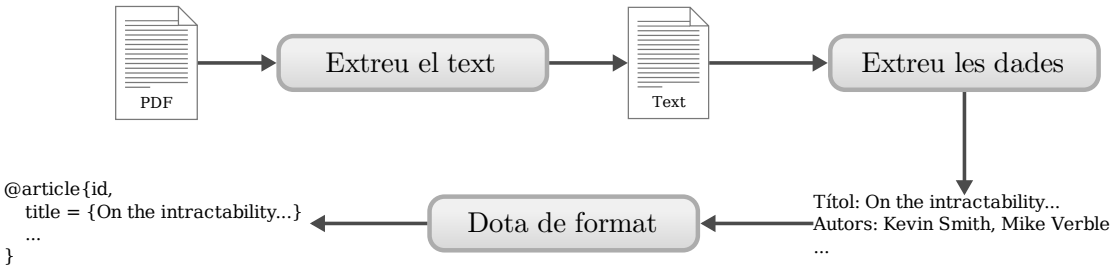


Figura 2.1: Primera idea per l'extracció de referències

Això presenta moltes limitacions. Per començar, hi ha la impossibilitat d'extreure informació que no es troba dins del text (e.g. el número de pàgines de l'article). Una altra és que la informació dins dels fitxers PDF no té una estructura prou clara com per poder distingir entre els diferents camp. Aquest segon problema es tracta amb més en detall al proper capítol.

La solució que hem trobat per poder tirar endavant consisteix en fer ús del que s'anomenen *biblioteques digitals*: grans bases de dades amb informació sobre articles. La majoria d'elles estan indexades pels principals cercadors disponibles a Internet. El diagrama actualitzat es mostra a la figura 2.2. Bàsicament, per cada fitxer l'aplicació:

1. Extreu el contingut en forma de text.
2. Genera un llistat de consultes a partir del text extret.
3. Obté els resultats d'aquestes consultes amb algun cercador (Google, Bing, etc.).
4. Mira d'extreure la referència de les pàgines retornades pel cercador.
5. Comprova que les dades de la referència realment corresponguin a l'article contingut pel fitxer.
6. Dóna format a les dades de la referència.

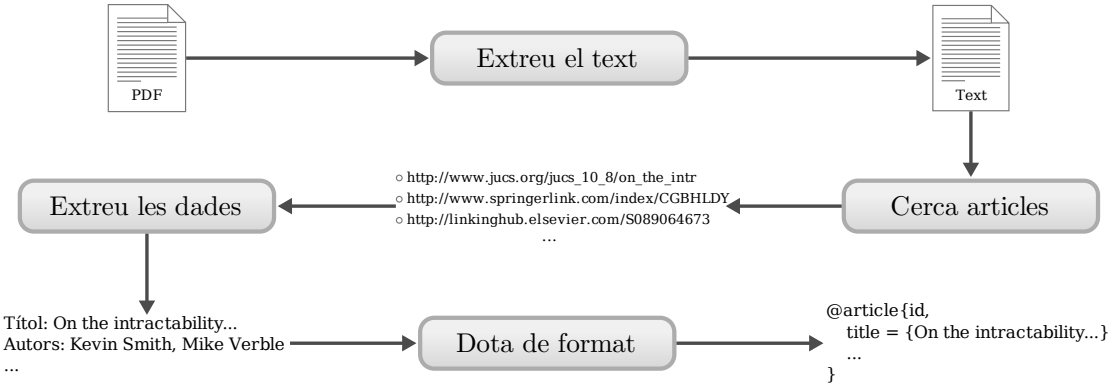


Figura 2.2: Esquema del procés a seguir per extreure una referència

El motiu que fa que aquest procediment es pugui implementar en el temps del que disposem és que les dades ofertes per les biblioteques digitals en els documents HTML està molt estructurada. Com és lògic, tenen la informació emmagatzemada en bases de dades, i a l'hora de mostrar-la als usuaris, la formaten de manera automàtica; sempre de la mateixa manera (amb algunes excepcions). Aquesta estructura fixa al llarg de les pàgines d'una mateixa biblioteca és el que realment simplifica la tasca d'extracció, ja permet generar regles que funcionin en un percentatge molt alt dels casos.

Així doncs, hem de fer una mica més de tomb per poder aconseguir els resultats desitjats, però els passos són més senzills d'aplicar. El problema que teníem al principi ha quedat dividit en tres problemes més petits:

- La cerca de les pàgines amb informació de l'article.
- Extracció de la referència d'aquestes pàgines.
- Generació de les regles d'extracció. (Les anomenarem *wrappers*)

Cadascuna d'aquestes parts es tracta per separat en els propers capítols.

2.3 BibTEX

Per poder entendre el context del projecte cal que descrivim l'eina de maneig de referències BibTEX i la sintaxi del llenguatge que utilitza. En el nostre cas farem servir aquest llenguatge com a format de sortida al generar els índexos bibliogràfics. Al llistat 2.1 es mostra un exemple d'una referència d'un article científic expressat en el format BibTEX:

```
@article{MoSh:27,
  title = {Size direction games over the real line},
  author = {Moran, Gadi and Shelah, M., Saharon},
  journal = {Israel Journal of Mathematics},
  pages = {442--449},
  volume = {14},
  year = {1973},
}
```

Llistat 2.1: Referència expressada en BibTEX

Alguns aspectes a comentar sobre l'exemple anterior:

- La primera línia conté el tipus de document i un identificador. El primer defineix els camps obligatoris que s'han d'especificar, i el segon ens permetrà citar a la referència des d'un document. En el nostre cas només ens interessen les referències de tipus *article* i haurem de definir, com a mínim, els camps *author*, *title*, *journal* i *year*, que són obligatoris.
- Es considera que el nom d'un autor o editor pot constar de quatre parts diferents: *First*, *von*, *Last*, *Jr.*. Es poden ordenar de diverses maneres, però nosaltres ho farem amb `<von> <last>, <middle>, <first>`. Cal separar múltiples noms amb la paraula `and`.
- L'últim camp d'una referència pot acabar o no amb una coma.

2.4 Disseny del sistema

Hem estructurat el codi en diferents mòduls independents; es llisten a continuació:

- *Raw Content Extraction* (`rce`): Agrupa totes les classes encarregades d'extreure el contingut dels documents PDF.
- *Information Retrieval* (`ir`): Encarregat de comunicar-se amb els diferents cercadors suportats per obtenir pàgines que contenen informació de la referència que volem extreure.
- *Information Extraction* (`ie`): Conté tot el codi que permet obtenir la referència a partir d'una pàgina HTML. A més, també és l'encarregat de generar nous *wrappers*.
- *References* (`references`): Per una banda fa un anàlisis sintàctic de les referències extretes per poder-les validar. Per l'altra, les transforma a BibTeX.
- Base de dades (`db`): Tal i com indica el seu nom, duu a terme els accessos la base de dades.
- *Main* (`main`): Enllaça tots els mòduls anteriors i proporciona punts d'entrada a la interfície d'usuari. Fa de façana del sistema.
- *Graphical User Interface* (`ui`): Interfície d'usuari més o menys amigable.

La figura 2.3 mostra com interaccionen entre ells. Els mòduls que s'encarreguen de les operacions més bàsiques són independents entre ells. El mòdul *main* utilitza aquestes operacions per definir el flux de la informació i ofereix serveis a la capa superior: la interfície d'usuari.

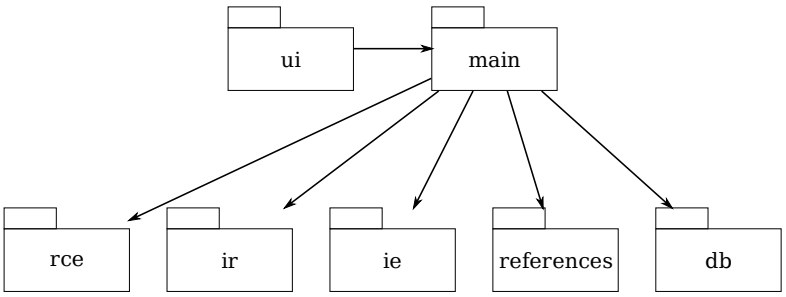


Figura 2.3: Mòduls del sistema

2.5 Llicència

L'aplicació es distribueix sota una licència lliure que permet als destinataris la possibilitat de copiar, modificar i redistribuir el codi. Concretament, hem escollit la *General Public License* (GPL) v.3, que requereix que qualsevol treball derivat també es distribueixi amb els mateixos permisos. El text complet d'aquesta llicència es pot trobar a [GPL] així com al fitxer `LICENSE` que acompanya al codi.

2.6 Llista de tasques

A continuació s'esmenten les tasques més importants que s'han dut a terme i elements que s'han implementat durant la realització del projecte. S'acompanyen amb una estimació del temps que ha estat necessari per completar-les, per donar una idea del pes de cadascuna d'elles:

Tasca	**Temps estimat**
Recerca i proves de concepte	**20 dies**
Configuració de la infraestructura	**4 dies**
Repositori (*GitHub*): Creació i estructura del projecte	1 dia
Build tracker	3 dies
Extracció del text dels PDF (`bibim.rce`)	**8 dies**
Cerca d'eines i comparació de resultats	5 dies
Codi de l'extracció	3 dies
Obtenció de pàgines amb la referència (`bibim.ir`)	**13.5 dies**
Proves amb diferents cercadors i *XGoogle* (veure l'apèndix A)	4 dies
Obtenir consultes del text	1.5 dies
Browser per obtenir pàgines de la Web	2 dies
Codi per la cerca de les consultes	5 dies
Ordenació resultats de la cerca	1 dia
Extracció d'informació (`bibim.ie`)	**14 dies**
Anàlisi de l'estructura d'algunes biblioteques digitals disponibles	5 dies
Regles a mà per extreure referències directament	3 dies
Regles a mà per extreure camps	2 dies
Validació de les referències extretes	4 dies
Generació automàtica de *wrappers* (`bibim.ie`)	**32.5 dies**
Extracció d'exemples	1.5 dies
Validació dels exemples extrets	0.5 dies
Wrapper genèric	0.5 dies
Generació automàtica de regles	27 dies
Camps de valor únic	17 dies
Camps multi valor	10 dies
Avaluació de les regles generades	3 dies
Tractament de referències (`bibim.references`)	**5 dies**
Parsing	2 dies
Generació i Format	2 dies
Creació índexs	1 dia
Base de dades (`bibim.db`)	**5 dies**
Disseny	1 dia
Mappers amb SQLAlchemy	4 dies
Mòdul principal (`bibim.main`)	**11 dies**
Escaneig de fitxers	1 dia
Threading	2 dies
Comunicació dels mòduls anteriors	8 dies
Interfície d'usuari (`bibim.ui`)	**4 dies**
Editor de referències	1.5 dies
Editor de *wrappers*	1 dia
Resta de vistes	1.5 dies

Proves	**12 dies**
Col·leccionar PDF, HTML i BibTeX de mostra	3 dies
Scripts per automatitzar	6 dies
Execució i anàlisi de resultats	3 dies

Capítol 3

Cerca de referències

3.1 Extracció dels continguts d'un PDF

El primer pas per poder obtenir la referència d'un article en un fitxer PDF és l'extracció del contingut d'aquest fitxer. Aquest és un dels aspectes que han influït més en l'enfocament que hem donat al sistema, pels motius que es descriuen a continuació.

Tal i com ja hem esmentat, en un principi, la solució que es va plantejar era intentar extreure la referència bibliogràfica d'un document directament del fitxer PDF del qual es disposa. Tot i les limitacions que això suposa, després de veure com queden els articles al convertir-los a text ens vam allunyar encara més d'aquesta idea.

```
Characterization and Armstrong Relations for Degenerate
   Multivalued Dependencies Using Formal Concept Analysis
Jaume Baixeries and Jos´ Luis Balc´zar e a
Dept. Llenguatges i Sistemes Inform‘tics, a Universitat
   Polit‘cnica de Catalunya, e c/ Jordi Girona, 1-3, 08034
   Barcelona {jbaixer, balqui}@lsi.upc.es

Abstract. Functional dependencies, a notion originated ...
```

Llistat 3.1: Text corresponent a la capçalera d'un article després d'haver-lo extret d'un PDF

Els llistats 3.1 i 3.2 mostren exemples de les capçaleres de dos articles diferents després d'haver extret el text del fitxer PDF en el que es trobaven. Com es pot veure, el resultat no té cap tipus d'estructura que pugui deixar intuir quina part del text correspon a cada fragment d'informació, sinó que és un conglomerat de totes dades. El primer article comença amb el títol i segueix amb els noms dels dos autors i la informació de la universitat. En canvi, el segon comença amb l'any, la conferència on s'ha presentat, títol i per cada autor es dóna informació diferent sobre la universitat. Si comencem a mirar més articles, el número de casos amb estructures diferents no para d'augmentar (hi ha més exemples a l'apèndix D).

```
2010 Second International Conference on Future Networks

Cloud Computing Research and Development Trend
```

```
Shuai Zhang Hebei Polytechnic University College of Science
    Hebei Polytechnic University NO.46 Xinhua West Street
    Tangshan 063009, Hebei Province China zhangshuai@heut.
    edu.cn Xuebin Chen Hebei Polytechnic University College
    of Science Hebei Polytechnic University NO.46 Xinhua
    West Street Tangshan 063009, Hebei Province China
    chxb@qq.comm
Abstract—With the development of parallel computing,
    distributed [...]
```

Llistat 3.2: Un altre exemple de text extret d'un PDF

Una vegada vistos els resultats, probablement quedin més clara la inviabilitat de la primera opció i per què hem decidit consultar la informació dels articles que hi ha disponible a la xarxa. De totes maneres, continua sent necessària l'extracció del text dels PDFs per tal de poder fer cerques, i cal veure com ho podem fer.

3.1.1 Dificultats

Tot hi haver-hi diverses utilitats que permeten l'extracció del contingut d'un fitxer PDF en forma de text pla o HTML, totes presenten problemes similars als de la llista següent:

- No extreuen bé els caràcters especials com ara Unicode o lligadures (e.g. *fi* es representa com un sol caràcter)
- Sub/Superíndexs: la majoria d'eines els extreuen com text que forma part de la paraula. Per exemple: *Joan*^{3} s'extreu com a *Joan3*
- Flux del text dins del fitxer: Hi ha casos en que el text es troba en diferents columnes i a l'hora d'agafar-lo, aquestes columnes o seccions no han d'estar mesclades.
- Fragmentació de paràgrafs: Relacionat amb el punt anterior. Hi ha ocasions on els paràgrafs es divideixen en un conjunt de línies segons com es troben posicionades dins del document. El text resultant conté salts de línia addicionals que s'han introduït per conservar les mateixes línies del document original, sense tenir en compte l'estructura lògica.
- Fitxers protegits dels quals no es pot extreure el contingut

Una altra situació en que no serem capaços d'extreure el text del PDF és en aquells casos que els fitxers enlloc de contenir text, contenen imatges amb el document escanejat i no han estat processats per cap programari de reconeixement de caràcters. Pel que hem vist, això sol passar sobretot per articles de fa uns quants anys.

3.1.2 Programari

El llistat de programari lliure disponible per a dur a terme l'extracció del contingut és força reduït i totes presenten alguns dels problemes (o tots) que acabem de comentar. Tot i així, hem tingut en compte diverses opcions abans d'escollir una biblioteca o aplicació d'extracció. Hem contemplat: *PyPDF*, *PDFMiner*, *PDFBox*, però finalment ens hem decantat per *xPDF*.

xPDF consisteix en un conjunt d'eines executables des de la línia de comandes que permeten extreure text i altres elements dels fitxers PDF. Es distribueixen sota la llicència GPL v.2 i hi ha binaris tant per Windows com per Linux (que també funcionen per Mac OS). El motiu principal pel qual hem escollit aquesta eina és que s'obtenen resultats relativament bons. En especial, és interessant el fet que no separa els paràgrafs en diferents línies i que en la majoria dels casos respecta el flux del text dins del document.

Pel que fa als caràcters especials, transforma bé les lligadures en múltiples caràcters, però té problemes amb la codificació Unicode. Donat que la majoria dels articles científics estan escrits en anglès, aquest és un problema que hem decidit obviar. Tal i com veurem, a no ser que l'article contingui un percentatge molt elevat d'aquest tipus de caràcters, serà igualment possible extreure'n la referència.

3.2 Consultes

El més important per poder cercar referències bibliogràfiques a Internet és ser capaços de generar consultes que retornin bons resultats. Per fer-ho, agafarem porcions del text extret amb l'eina *xPDF* i les utilitzarem per obtenir resultats que hi coincideixin exactament.

Una primera idea pot consistir en cercar segons el títol de la publicació de la qual volem informació. El problema és que bona part dels resultats corresponen a pàgines que fan referència a aquesta publicació, però que no en donen gaires detalls. Agafant la resta d'informació de la capçalera (e.g. autors, revista) les consultes encara seran menys restrictives i retornaran resultats pitjors. Per una altra banda, si intentem fer consultes a partir del contingut del propi article ens trobem amb que en molts casos, els cercadors no el tenen indexat.

Una tercera opció, que és la que utilitzem, consisteix en generar les consultes a partir del resum o *abstract* que acompanya a la majoria d'articles i que també acostuma a aparéixer a les pàgines que contenen la referència. Però com podem saber quina part del text que hem extret correspon al resum? Tot i que en moltes vegades el primer paràgraf va precedit de la paraula *Abstract*, també n'hi ha moltes altres en que va precedit d'una paraula completament diferent (e.g. resum o *summary*) o bé per cap. Per tal que el sistema sigui el més general possible, enlloc de fixar-nos en paraules concretes fem servir una expressió regular molt simple que permet trobar cadenes amb un número de paraules determinat.

Un dels trets característics de les capçaleres dels articles una vegada n'hem extret el text és que contenen un nombre elevat de símbols especials. Això ens pot ajudar a distingir entre les parts corresponents a la capçalera i el resum. L'expressió regular que obté les consultes és: `([\w()?!]+[ ]){min,max}` i agafarà seqüències de *min* a *max* paraules separades per un espai i formades per caràcters alfanumèrics i un nombre limitat de símbols. Els paràmetres *min* i *max* són configurables. Òbviament, les consultes que ens dóna aquesta expressió no sempre són bones i per tal de contrarrestar aquests errors, en generem vàries i les anem utilitzant mentre no s'obtinguin resultats satisfactoris. De totes maneres, tal i com es pot veure al capítol 6, no és necessari ni generar moltes consultes ni cal que aquestes siguin gaire llargues.

A continuació es llisten cinc consultes extretes d'un article d'exemple. Noteu que s'envolten de cometes dobles, la forma habitual d'indicar als cercadors que les coincidències han de ser exactes.

- "are known to admit interesting characterizations in terms of Formal"
- "natural extensions of the notion of functional dependency are the"
- "We propose here a new Galois"
- "which gives rise to a formal concept lattice corresponding precisely"
- "the degenerate multivalued dependencies that hold in the relation"

En molts casos, l'expressió regular anterior també dóna coincidències pel títol de l'article. Per evitar el problema que hem descrit, hi ha definit un altre paràmetre que estableix el nombre de consultes a saltar-se des del principi de l'article. És una manera rudimentària d'aconseguir-ho, però funciona la majoria de vegades.

3.3 Cercadors

El següent pas després d'haver obtingut un conjunt de consultes és utilitzar-les amb un cercador per tal d'obtenir pàgines amb informació de la referència que volem aconseguir. Al capítol d'introducció, hem esmentat que hi ha cercadors com ara *Google Scholar* o *Microsoft Academic Search* on els resultats només corresponen a publicacions. En un principi ens va semblar raonable intentar fer ús d'aquests serveis per poder aconseguir els nostres objectius. El problema és que no tenen cap API publicada que permeti fer consultes automàtiques des d'aplicacions de tercers. Tot i que hi ha solucions i *workarounds*, van en contra dels termes i condicions i els servidors bloquegen massa consultes seguides. Per tant, hem descartat aquesta opció.

Així doncs, ens quedem amb els cercadors habituals; hem preparat la nostra aplicació per tal d'utilitzar les APIs de *Google*, *Yahoo* i *Bing*. Tots tres cercadors retornen els resultats en format JSON. Els principals inconvenients són que retornen qualsevol tipus de pàgina i que no tenen indexades algunes blioteques digitals. Tot i així, també podem aconseguir bons resultats amb l'ús de les consultes adequades. Al capítol de resultats (6.1) hi ha una comparativa.

3.3.1 Ordenació de resultats

La majoria de vegades, no ens convindrà l'ordre dels resultats donat pels diferents cercadors sinó que voldrem processar les pàgines segons aquelles per les quals tenim regles d'extracció. És per això que un cop hem consultat al cercador, comprovem si tenim regles per alguna de les pàgines resultants i, en cas afirmatiu, la movem a dalt de tot de la llista.

En cas d'haver-hi múltiples pàgines de les quals sabem extreure les dades, ho farem segons una puntuació assignada a les regles de les que disposem. Aquest tema de les puntuacions es tracta amb més detall al capítol sobre generació de *wrappers* (5).

3.3.2 Altres Ajustaments

Depenent de l'estructura del contingut dels fitxers dels que disposem, la qualitat dels resultats obtinguts amb els cercadors pot variar considerablement. Això suposa la necessitat d'ajustar alguns paràmetres per tal de poder adaptar el sistema a l'ús de cadascú. A la secció sobre la generació de consultes (3.2), ja hem comentat la possibilitat d'ajustar el mínim i màxim de termes a cercar, però hi ha altres opcions que es poden configurar.

En algunes ocasions, es dóna el cas que la consulta generada no és prou restrictiva, ja sigui perquè no és prou llarga o bé perquè està formada per paraules molt generals. Al cercar amb aquestes consultes s'obté una llarga llista de resultats, la majoria dels quals no tenen res a veure amb la informació que estem buscant. Per contrarestar-ho, hi ha la possibilitat d'indicar al sistema que ometi els resultats i provi amb la següent consulta. A l'hora d'assignar el valor d'aquest paràmetre, també s'haurà de tenir en compte el tipus d'articles dels que es vol informació. Per exemple, els articles populars segurament tindran un número de coincidències rellevants gran i, per tant, haurem d'assignar un valor relativament alt, ja que un valor baix farà que descartem resultats bons. En canvi, per articles poc corrents, ens interessarà el contrari.

Per una altra banda, hi ha ocasions en que els cercadors tenen tendència a retornar resultats que, tot i coincidir amb la consulta que li hem donat, corresponen a una pàgina que no ens aporta massa informació. Per tal d'ajudar a l'aplicació a descartar resultats dolents, podem indicar-li pàgines que volem ometre a partir d'una llista negra. Per exemple, sabem que les pàgines sobre els autors de la biblioteca digital *ACM Portal* contenen un llistat de tots els articles d'un mateix autor, però que no contenen suficient informació com per extreure referències. En aquest cas voldrem descartar els resultats que comencen per http://portal.acm.org/author_page.cfm.

3.4 *Multithreading*

Un dels inconvenients més grans que implica el fet d'haver d'accedir a Internet, és que el temps perdut esperant dades és molt alt. Per reduir-lo, s'ha estudiat la possibilitat d'utilitzar diferents fils d'execució per fer més d'una consulta de forma més o menys simultània. La taula següent mostra una comparativa del temps necessari per obtenir mútliples pàgines web de forma seqüencial o bé utilitzant fins a cinc fils d'execució diferents. Les pàgines pàgines corresponen a consultes aleatòries a *Google* per evitar l'efecte dels *proxies* i *caches*.

2 pàgines		5 pàgines		10 pàgines		20 pàgines	
Seq.	5 Threads	Seq.	5 Threads	Seq.	5 Threads	Seq.	5 Threads
0.9010	0.5481	2.1830	0.6612	4.3153	1.5914	7.9295	2.5949
0.7467	0.3795	2.1558	0.7441	4.3186	1.2311	8.5483	2.1958
0.7678	0.5641	2.0645	0.5383	9.2930	1.4415	8.7202	2.5749
0.7421	0.3876	2.0684	0.8551	4.9859	1.5294	8.4732	2.2841
0.9674	0.5477	2.1510	0.8550	5.3600	1.3116	9.2901	2.2257
Mitjana:							
0.8250	0.4854	2.1246	0.7307	5.6546	1.4210	8.5923	2.3751
Guany:							
-44.96%		**-65.6%**		**-74.87%**		**-72.35%**	

Tot i que aquestes proves no són riguroses, sí que són suficients per poder-nos fer una idea força clara sobre la millora que s'obté utilitzant múltiples fils respecte no fer-ho. Pere exemple, per obtenir 5 pàgines amb un sol fil només es tarda lleugerament menys que per obtenir-ne 20 amb 5 fils.

Pel que fa a la forma d'implementar-ho, hem creat un *pool* amb un número màxim configurable de fils d'execució que es van reutilitzant mentre queden referències per extreure. Bàsicament, tenim una cua amb les rutes als fitxers PDF i una cua de sortida amb el resultat d'extreure les referències. Cada *thread* va processant fitxers de la cua d'entrada mentre aquesta no és buida. El número de fils s'haurà d'ajustar segons del tipus de connexió del que es disposi.

Capítol 4

Extracció de referències

Un cop hem aconseguit trobar pàgines que contenen informació de l'article pel qual volem generar la referència bibliogràfica, és moment d'extreure aquesta informació i formatar-la.

4.1 *Wrappers*

En el nostre context, anomenarem *wrapper* a una classe que implementa una sèrie de mètodes establerts i que, a partir d'un text o document d'entrada, permet extreure'n certa informació. Podem imaginar-ho com un filtre que només ens deixa veure una part del document que ens interessa.

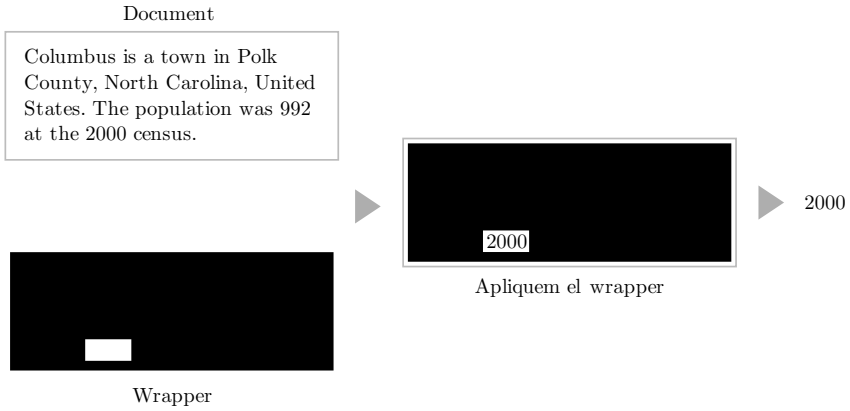


Figura 4.1: Exemple de la funció d'un *wrapper*

Internament, consisteixen en un seguit de regles especialitzades en extreure les dades de documents estructurats d'una forma concreta. A la nostra aplicació tindrem els dos tipus de *wrappers* que es descriuen a continuació, que es diferencien entre ells per la funció que realitzen.

4.1.1 *Reference Wrappers*

Aquest tipus de *wrappers* extreuen el text corresponent a una referència sencera dins de les pàgines HTML dels resultats. El gran avantatge que tenen és que habitualment permeten obtenir

molta informació i amb una confiança molt més gran. Ara per ara, el sistema només suporta referències BibTeX, però es podria ampliar amb qualsevol altre format.

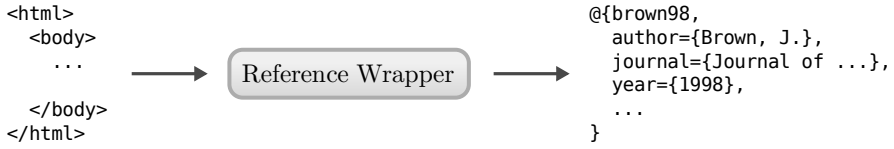


Figura 4.2: Esquema de funcionament dels *reference wrappers*

El principal problema és que s'han d'implementar manualment ja que moltes vegades el text de les referències no es troba a la mateixa pàgina retornada pels cercadors, sinó que cal seguir algun enllaç o realitzar altres accions abans d'arribar-hi, cosa que complica força la generació automàtica. A més, el fet que d'aquests *wrappers* només en necessitem un per cada biblioteca, fa que no s'hagi considerat oportú automatitzar-ne la generació ja que (de moment) no suposa un estalvi de recursos prou gran.

Un altre inconvenient d'aquest tipus de *wrappers* és que hi ha moltes biblioteques digitals que no ofereixen les referències en BibTeX sinó en altres formats com ara *RIS*, *MODS*, etc. La llista de formats que s'ofereixen depèn força del camp de coneixement en què s'especialitza la biblioteca.

Respecte a la manera d'implementar-los, el primer que cal fer és estudiar el funcionament de cada biblioteca i aplicar una mica d'enginyeria inversa. Per exemple, en algunes ocasions podem aconseguir alguna drecera per arribar a les referències a partir de les direccions retornades pels cercadors. Un dels casos més senzills és el de la biblioteca digital *ScientificCommons* on per obtenir el codi BibTeX només hem de canviar l'adreça retornada pel cercador:

```
http://en.scientificcommons.org/32119993
```

```
http://en.scientificcommons.org/export/bibtex/32119993
```

Altres vegades, obtenir la referència és més complicat, i és necessari construir la URL a partir de paràmetres d'una crida a una funció *JavaScript*, com passa per la biblioteca *ACM*; o bé a partir de camps d'un formulari, cas de *ScienceDirect*, etc.

4.1.2 *Field Wrappers*

A diferència dels que acabem de veure, aquest tipus de *wrappers* s'especialitzen en extreure únicament un dels camps de la referència cada vegada. Per tant, se'n necessita un per cadascun dels camps que volem obtenir per cada biblioteca d'articles que vulguem suportar. El diagrama següent mostra la diferència respecte els de la figura 4.2:

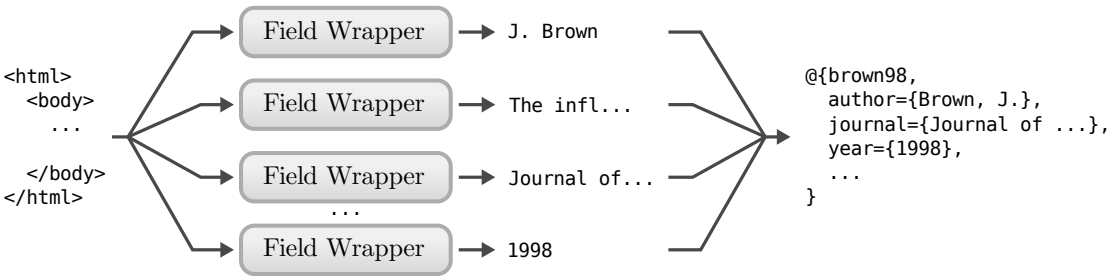


Figura 4.3: Esquema de funcionament dels *field wrappers*

Internament, aquests tipus de *wrappers* consisteixen en un llistat de regles que cal aplicar en un ordre determinat i que, si tot va bé, dónen com a resultat el valor que volem aconseguir. Aquestes regles estan connectades en cascada, de manera que el tipus de sortida d'una regla ha de ser vàlida per l'entrada de la següent.

Per assegurar-nos que l'aplicació pot acomplir els objectius proposats, en un principi vam començar a definir algunes d'aquestes regles manualment, mirant de fer-les prou generals com per poder-les reutilitzar per múltiples biblioteques. És fàcil veure que aquesta és una tasca que consumeix molt temps. A més, s'ha de tenir en compte que els resultats obtinguts es veuen afectats per qualsevol canvi en l'estructura de les pàgines font. Cada vegada que hi ha un redisseny, s'han d'actualitzar les regles o bé crear-les de nou. Per tant, vam decidir dedicar la resta de projecte en trobar la manera de generar aquest tipus de *wrappers* de forma automàtica. La forma de fer-ho es descriu al capítol 5.

De totes maneres, cenyint-nos al procés d'extracció, els passos que se segueixen són:

- Donat el resultat web d'un article mirem si disposem de *wrappers* per algun dels camps a la base de dades de l'aplicació.
- De ser així, per cada camp obtenim els millors segons una puntuació.
- Apliquem el *wrapper*
- Si el resultat es considera vàlid, passem al següent camp. Altrament, provem amb un altre *wrapper*.

Com es pot deduir, no serà necessari tenir *wrappers* que funcionin en tots els casos sinó que només caldrà tenir-ne uns quants que ho facin en un percentatge prou elevat. Això facilita l'extracció per aquelles biblioteques digitals que varien l'estructura de la informació en algunes ocasions.

4.2 Validació de referències

Després d'obtenir les dades dels documents HTML amb els mètodes indicats, ens interessa validar-les per diversos motius:

- Continuar provant més resultats i regles disponibles abans de donar l'extracció per finalitzada.

- Indicar a l'usuari que sospitem que algun dels camps no és correcte.
- Modificar la puntuació dels *wrappers* per tal d'elegir els millors en properes extraccions.
- Evitar utilitzar referències incorrectes a l'hora de generar nous *wrappers*

El procés de validació de les dades depèn de cada camp i és totalment ajustable. Dins del fitxer de configuració de l'aplicació, es pot establir, per cada camp, com s'ha de validar i un pes sobre la validesa total de la referència. Pel que fa al primer paràmetre, tindrem diferents tipus de validadors:

- *WithinTextValidator*: Basa la validació en comprovar si la cadena extreta es troba dins del text extret de l'article PDF.
- *PersonValidator*: Semblant a l'anterior, però ho comprova pels diferents noms de les persones.
- *RegexValidator*: Mira que el text extret coincideixi amb una expressió regular que també s'inclou al fitxer de configuració. Útil per aquells casos on el camp no es troba dins del document de l'article (e.g. pàgines, issn).

El pes correspon a un nombre en coma flotant que ens permet donar més importancia a certs camps com ara el títol o nom dels autors a l'hora d'establir com de vàlida és una referència. La suma total dels pesos ha de ser igual a 1.

Per configurar la validació s'utilitza el valor `field_validation` dins del fitxer de configuració. La sintaxi per definir com validar cadascun dels camps és `<field>; <weight>; <validator>[; <validator params>]`. A continuació es mostra un exemple:

```
field_validation=
    title;    0.3; WithinTextValidator
    journal;  0.2; WithinTextValidator
    author;   0.2; PersonValidator
    pages;    0.1; RegexValidator;     \d+(?:\ ?[-,]?\ ?\d+)?
    issn;     0.0; RegexValidator;     (\d{4}-\d{3}(\d|X))
    ...
```

Llistat 4.1: Configuració de la validació de referències

Per anar bé, cal especificar un validador per cadascun dels camps, fins i tot per aquells als que es dóna pes 0.0. Això garantitza que la puntuació dels *wrappers* per aquests camps es continuï actualitzant tot i no tenir-los en compte a l'hora de considerar la referència com a vàlida.

Parsing de referències

Per poder validar les referències extretes amb dels *reference wrappers*, és necessari analitzar-les sintàcticament i obtenir-ne els diferents camps per separat. Per aquest motiu, l'aplicació disposa d'un *parser* de referències en format BibTeX.

4.3 Format de referències

Tal com s'ha indicat al diagrama de la figura 2.2, un cop tenim les dades extretes cal donar-los-hi format per poder-les exportar en BibTeX. Com a detalls de la implementació, només comentar que tenim un jerarquia de classes anomenades *generadors* que, guiades per una classe formatadora (`Formatter`) permetran generar les referències en el format desitjat.

4.4 Emmagatzematge

Totes les referències extretes s'emmagatzemen a la base de dades de l'aplicació per poder-les utilitzar en un futur tant per exportar-les com per regenerar els *wrappers* amb què s'han obtingut en primer lloc. A part de la referència en si, també es desa la consulta extreta del PDF per cercar a Internet i el resultat de la pàgina de la qual s'ha extret la referència.

En referència al disseny de la base de dades, a l'apèndix C.3 hi ha un diagrama detallant les diferents taules, camps i relacions.

Capítol 5

Generació de *Wrappers*

En aquest capítol s'expliquen les tècniques utilitzades per la generació automàtica de regles d'extracció de la informació. A grans trets, el procediment a seguir consisteix en agafar exemples de referències i mirar on es troba cada camp dins de la pàgina que conté la referència, per tal d'aplicar-ho en altres pàgines amb la mateixa estructura. Cal comentar que com que els resultats no seran perfectes, l'usuari sempre podrà corregir els *wrappers* generats per maximitzar la correctesa de les dades extretes en cada cas.

5.1 Obtenció d'exemples

Per poder generar *field wrappers*, es necessiten exemples del camp que es vol extreure. Un exemple està format pel valor que volem obtenir i pel context en què es troba aquest valor. S'obtenen a partir de les referències emmagatzemades a la base de dades de l'aplicació, que poden haver estat importades d'un fitxer `.bib` o bé extretes anteriorment utilitzant *wrappers* que amb el temps han deixat de funcionar. Totes les referències hauran de tenir associada una URL que apunti a una pàgina que contingui la informació de la referència. En el cas de les que s'han obtingut automàticament, aquesta adreça es desa durant l'extracció.

A l'inici, el context dels valors dels nostres exemples és el codi HTML que conté la referència. Per tant, el primer pas consisteix en obtenir aquestes pàgines d'Internet. Tot i que per cadascun dels camps es generen *sets* d'exemples diferents, les pàgines són les mateixes i només es descarreguen una sola vegada. Tot i així, com que totes es trobaran a la mateixa biblioteca i les peticions les procesa el mateix servidor, l'aplicació mira d'evitar bloquejos esperant uns segons entre petició i petició. Un cop tenim els exemples generats, mirem si l'HTML conté la informació que volem extreure i si no és així, l'exemple es marca com a invàlid per tal de no tornar-lo a fer servir en un futur.

Abans de fer-lo servir, el codi HTML es neteja per tal d'eliminar-ne els comentaris, salts de línia, i altres elements que no ens interessen, com ara codi *JavaScript* o etiquetes d'estil. El número d'exemples a tenir en compte per generar els *field wrappers* és configurable, però com a mímin se'n necessiten dos que siguin vàlids.

5.2 Generació automàtica de regles

Al capítol anterior s'ha explicat que els *field wrappers* estan formats per una llista de regles que s'han d'aplicar en un ordre concret per tal de poder extreure el camp que volem. Per nosaltres una regla està composta per un patró i un procediment que aplica aquest patró; en tenim de diferents tipus segons les dades d'entrada i sortida que reben i retornen. Les regles es connectaran en cascada, de manera que és necessari que el tipus de sortida d'una sigui vàlida per l'entrada de la següent.

Els detalls respecte com es creen varien depenent de cada tipus de regla, però els dos passos bàsics que se segueixen són els següents. Donat un *set* d'exemples:

1. Es generen regles per un dels exemples.

2. Es fusionen les regles obtingudes amb les anteriors.

Donat que volem extreure informació de documents HTML, les regles mínimes que necessitem són: una per localitzar l'etiqueta que conté el valor que ens interessa i una altra per extreure'n aquest valor entre tot el text que pugui acompanyar-lo dins de la mateixa etiqueta. Les hem anomenat *path rules* i *regex rules*, respectivament.

5.2.1 *Path Rules*

Tal i com acabem de comentar, són les regles que permeten localitzar trossos d'informació dins de la pàgina. Els patrons d'aquest tipus de regles consisteixen en una mena de ruta que permet arribar a l'etiqueta HTML que conté el valor del camp. Tenen l'aspecte que es mostra a continuació, formats per una expressió regular i una llista de *triplets* compostos pel nom de l'etiqueta, els seus atributs i la posició respecte els seus *germans*:

```
['(\d{4}-\d{3}(\d|X))', (u'table', {u'width': u'100%'}, 7),
    (u'tr', {}, 0), (u'td', {}, 0)]
```

Per generar aquests patrons, cerquem el valor que volem extreure dins de l'HTML i anem pujant l'arbre sintàctic que descriu el document fins arribar a un antecessor amb nom d'etiqueta i atributs únics. Per cada element en guardem les seves característiques de manera que després puguem recórrer el mateix camí a l'inrevés.

Pel que fa al número de *sibling* o germà, només l'utilitzem per agilitzar el procés de cerca de la informació a l'hora d'aplicar la regla. En un principi havíem pensat distingir elements segons la seva posició respecte als germans, però això no funciona en aquells casos on múltiples pàgines d'una biblioteca digital mostren la mateixa informació en un ordre diferent. Per exemple, si tinguéssim els dos fragments de codi següents, al cercar el número de volum acabaríem amb les rutes simplificades: `[(table, 0), (tr, 1), (td, 0)]` i `[(table, 0), (tr, 0), (td, 0)]`. En realitat, però, ens interessaria combinar les dues rutes en una de sola i que ens permetés obtenir l'element correcte en qualsevol dels casos.

```
<table>
  <tr>
    <td class='label'>
      Year:
    </td>
    <td class='value'>
      1985
    </td>
  </tr>
  <tr>
    <td class='label'>
      Volume:
    </td>
    <td class='value'>
      323
    </td>
  </tr>
</table>
```

```
<table>
  <tr>
    <td class='label'>
      Volume:
    </td>
    <td class='value'>
      468
    </td>
  </tr>
  <tr>
    <td class='label'>
      Issue:
    </td>
    <td class='value'>
      3
    </td>
  </tr>
</table>
```

A l'hora de fusionar dos patrons, només ho fem si coincideixen en la llargada, nom d'etiqueta i atributs dels elements de la ruta. És a dir, l'únic que es modifica és el número de germà: si les rutes tenen el mateix, el deixem tal i com està; si difereixen, el substituïm amb el valor -1. Quan dos o més patrons no es poden fusionar, el que fem és crear *wrappers* diferents amb cadascun d'ells i a l'hora d'aplicar-los, s'escull aquell que en teoria funciona millor.

Mentre apliquem el patró, si veiem un -1 al número de *sibling*, enlloc d'escollir el següent element directament, fem una cerca de tots aquells al nivell actual de l'arbre i que compleixen la resta de condicions. Això implica que enlloc d'obtenir un sol element com a resultat, en molts casos en tindrem més d'un. Per exemple, la figura següent mostra el resultat d'aplicar la regla `[('table',5), ('tr',-1), ('td', 1)]` en un document HTML. En aquest cas obtenim els tres elements subratllats. Com pot saber l'aplicació quin dels camps volem?

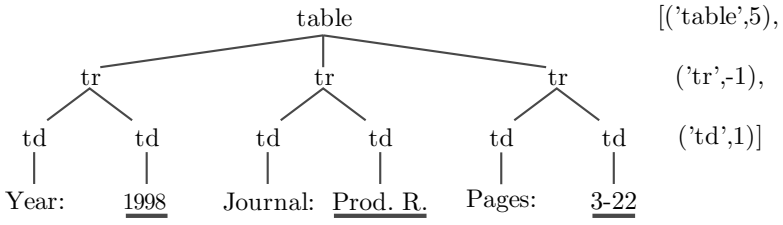


Figura 5.1: Exemple dels resultats a l'aplicar una *path rule*

Aquí és on entra en joc l'expressió regular de l'inici del patró. Amb aquesta expressió, podem ajudar a l'aplicació a descartar des d'un principi aquells elements que estem segurs que no ens interessaran. En l'exemple anterior, per obtenir l'any, podem tenir definida l'expressió `(\d{4})` i així quedar-nos només amb el primer element. El patró de la regla en realitat serà: `[(\d{4}), ('table',5), ('tr',-1), ('td', 1)]`. Dins del fitxer de configuració es poden definir expressions regulars per cada camp. Com és lògic, en alguns casos ens serà més útil que en altres.

Si estem intentant extreure camps que solen tenir sempre la mateixa forma com ara anys o bé codis com l'ISSN d'un article, serà fàcil indicar-li una expressió. En canvi, pels valors que no tenen cap tipus d'estructura, com ara els noms de revistes, no ens quedarà cap més remei que deixar-ho amb el valor per defecte `(.*)`

Un altre avantatge de tenir aquesta expressió regular és que ens servirà per corregir el comportament dels *wrappers* una vegada generats. Per exemple, si veiem que l'any de publicació no s'extreu correctament per una biblioteca determinada amb l'expressió `(\d{4})`, podem restringir més el número de coincidències canviant-la per `(Year\:\ \d{4})`

5.2.2 *Regex Rules*

Aquestes regles permeten extreure un camp quan el valor d'aquest es troba dins d'un element del document HTML acompanyat de més text. Reben una o més cadenes de caràcters com a entrada i els hi apliquen una expressió regular generada automàticament per tal de quedar-se només amb una porció la cadena.

Per generar les expressions regulars, s'agafa el text contingut als elements HTML que ens proporcionen les *path rules* i es fa una substitució del valor que volem obtenir per l'expressió `(.*)`. Per exemple, si volguéssim obtenir el número de pàgines d'una pàgina que conté la primera línia del llistat següent, començaríem posant els caràcters d'escapament necessaris i substituint el valor `1204-1209`.

```
Vol. 27, No. 6. (1 April 2006), pp. 1204-1209.
```

```
Vol\.\ 27\,\ No\.\ 6\.\ \(1 April 2006\)\,\ pp\. (.*)\.
```

De moment, com que només hem generat l'expressió amb una sola regla, no és prou general per extreure els camps de totes les pàgines. A a mesura que es provi amb més exemples, els trossos de text que varien aniran desapareixent. Suposem que ens arriba un altre exemple i la *path rule* ens retorna la primera línia següent, per la qual també en generem l'expressió.

```
Vol. 24, No. 1. (2 March 1999), pp. 332-344.
```

```
Vol\.\ 31\,\ No\.\ 1\.\ \(2 March 1999\)\,\ pp\. (.*)\.
```

A simple vista veiem que es tracta de la mateixa part del document HTML que la de l'exemple anterior, i que hauríem de poder aplicar la mateixa expressió regular en els dos casos per tal d'obtenir el número de pàgines. La tècnica que fem servir per decidir si hem de fusionar dues expressions és molt simplista, tan sols avaluem la similaritat de les expressions i comprovem si supera un llindar establert. De ser així, obtenim els blocs no coincidents de les dues cadenes de caràcters i els substituïm per `(?:.*)` de manera que s'accepti qualsevol cadena, però que es descarti l'hora d'aplicar l'expressió.

Les comparacions de seqüències es fan amb el mòdul `difflib` de la biblioteca estàndard de *Python*. Internament s'utilitza una variant de l'algorisme de reconeixement de patrons de Ratcliff/Obershelp, que mira de trobar la subseqüència coincident més llarga tot eliminant els elements no desitjats [Pyta], [RM88]. En l'exemple, el resultat de fusionar les dues expressions és:

```
Vol\.\ (?:.*)\,\ No\.\ (?:.*)\.\ \((?:.*)r(?:.*)\)\,\ pp\.
    (.*)\.
```

Com es pot veure, encara ha quedat la lletra *r* entre dos blocs a descartar. Durant la fusió d'expressions el sistema també aplica heurístiques que ajuden a generalitzar més ràpid. Per exemple, quan troba blocs coincidents de llargada 1, formats per una lletra o un número, els elimina. Per altra banda, també s'agrupen els blocs consecutius a descartar. D'aquesta manera, l'expressió final resultant queda:

```
Vol\.\ (?:.*)\,\ No\.\ (?:.*)\.\ \((?:.*)\)\,\ pp\. (.*)\.
```

El mètode que acabem de descriure per generar les expressions regulars no està exempt de problemes. Ja hem vist la necessitat de tenir bons exemples per poder generar expressions prou generals. Una altra situació en que ens podem trobar és que dues seqüències amb la mateixa estructura siguin massa diferents com per decidir fusionar-les. Quan això passa, acabarem amb múltiples expressions molt específiques. Per exemple, "`ISSN: 0000-0000, Issue: (.*)`" i "`ISSN: 1111-1111, Issue: (.*)`" tenen una similaritat de 0.71. Si el llindar per la fusió fos 0.75, aquests dos patrons es quedarien com estan i acabaríem generant dos *wrappers* diferents que segurament no funcionarien per cap altre cas. Veiem doncs, que aquesta tècnica és molt sensible als exemples.

Es podrien aplicar moltes millores per poder convergir a una expressió més general fent servir pocs exemples. En un principi generàvem les expressions de forma creixent, agafant el valor a extreure, i mirant els primers caràcters de la vora (aquesta és una simplificació de com ho feia el sistema *WHISK* [Sod99]). Vam abandonar la idea ja que teníem problemes quan la informació del context també variava entre pàgina i pàgina. En perspectiva, sembla que aquesta idea podria ser aplicada a les expressions que es generen actualment i així acabar amb expressions més curtes i molt mes generals. Si ho apliquessim amb els exemples que hem vist en aquesta secció, les expressions resultants serien:

```
\ pp\. (.*)\.
```

```
Issue:\ (.)
```

Tot i els problemes, cal tornar a remarcar que l'aplicació permet, en tot moment, l'edició de regles per part de l'usuari. De manera que si una expressió regular no s'ha acabat de generar prou bé, sempre es pot modificar amb un esforç mínim.

5.2.3 Regles multi valor

Alguns camps com ara els autors o editors tenen múltiples valors i per poder-los extreure per separat, hem creat variants de les regles que acabem de descriure. Suposem que múltiples valors corresponenen al mateix camp si compleixen alguna de les dues condicions següents:

- Es troben en elements HTML diferents, però són germans, cosins o tenen vincle parentiu similar. És a dir, quan les rutes per arribar als valors estan formades pels mateixos elements, però en posicions diferents. Aquesta condició permet extreure valors quan es troben tant en llistes com en taules:

```
<ul>
  <li>
    Liu Jing
  </li>
  <li>
  Li Jiandong
  </li>
  <li>
    Chen Yanhui
  </li>
</ul>
```

Germans

```
<table>
  <tr>
    <td>Autors:</td>
    <td>Liu Jing</td>
  </tr>

  <tr>
    <td></td>
    <td>Li Jiandong</td>
  </tr>
</table>
```

Cosins

- Es troben dins el mateix HTML amb un o més separadors entre ells. Per exemple, els tres autors següents estan separats per les cadenes ", " i " and ".

```
<td>Liu Jing, Li Jiandong and Chen Yanhui</td>
```

Llistat 5.1: Exemple múltiples valors

De la mateixa manera que pels camps d'un sol valor, també necessitem obtenir els elements HTML que contenen les dades a extreure. La tècnica és molt similar que la que hem explicat a la secció 5.2.1, l'única diferència és que en les ocasions en que els diferents valors es troben en elements diferents, cal fusionar-los.

Pel que fa a les *regex rules*, el seu equivalent pels *wrappers* multi-valor, és més diferent. Les substituïm per *Separator rules*, *Multi-value regex rules* i *Person rules*:

Separator rules

Són les encarregades de dividir el text d'un únic element en múltiples valors segons una sèrie de separadors. El seu patró consisteix en una llista amb aquests separadors.

El procediment per per generar-les és força trivial. Simplement s'agafa el text de l'element retornat per les *path rules*, i se substitueixen els valors que es volen obtenir per alguna cadena que no existeixi dins del text (e.g. l'expressió `(.*)`).

```
Liu Jing, Li Jiandong, Wai Kwan and Chen Yanhui
```

```
(.*), (.*), (.*) and (.*)
```

Un cop fet això, s'agafen totes les subcadenes o *separadors* que han quedat entre els elements substituïts i les fusionem. Aquest procés de *merging* és el mateix que el que se segueix per les expressions de les *regex rules*.

```
[', ', ', ', ' and ']
```

```
[', ', ' and ']
```

Multi-value regex rules

Funcionen de la mateixa manera que les *regex rules* de l'apartat anterior, però amb la diferència que reben i retornen múltiples valors. L'expressió regular del patró s'aplica per cadascun dels valors de l'entrada. Això permet extreure els valors de forma correcta quan tots, o alguns d'ells, van acompanyats d'altra informació. Per exemple, en el cas de rebre els valors:

```
Liu Jing     – Department of Electrical Engineering
Li Jiandong – Department of Applied Physics
Chen Yanhui – Department of Electrical Engineering
```

Es generarà el patró `(.*)\ \-\ Department\ of\ (?:.*)`, de manera que es podran obtenir els noms dels tres autors.

Person rules

Només s'apliquen en el cas que els valors del camp siguin noms de persones, com ara els autors i els editors. Aquestes regles reben el nom complet d'una persona i s'encarreguen de separar-lo en diferents parts. Es tenen en compte les parts: *first name*, *middle name* i *last name*. El fet de tenir els noms separats d'aquesta manera permet que a l'hora de formatar la referència (en el nostre cas en BibTeX) tots els noms tinguin la mateixa estructura. A continuació es mostren exemples de com quedarien dos noms:

```
David P. Bartel

{u'first_name':
   u'David',
 u'middle_name':
   u'P.',
 u'last_name':
   u'Bartel'}
```

```
James Green

{u'first_name':
   u'James',
 u'middle_name':
   u'',
 u'last_name':
   u'Green'}
```

Les regles que acabem de llistar s'han d'aplicar en aquest mateix ordre i, igual que amb la resta de *wrappers*, estaran connectades en cascada:

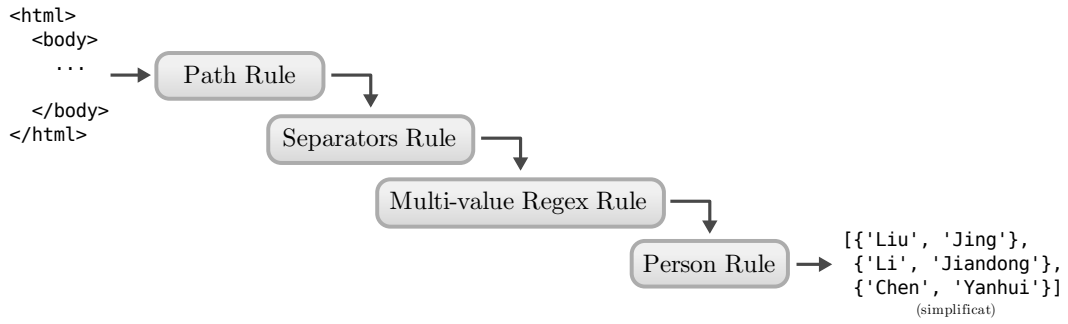


Figura 5.2: Exemple d'extracció de múltiples noms

5.3 Avaluació dels *wrappers*

Una vegada hem generat el conjunt dels *wrappers* possibles per a cadascun dels camps, cal que avaluem quins d'ells funcionen millor. Primer, els avaluem amb els mateixos exemples que hem fet servir per la generació. Això ens donarà una idea o confiança sobre com poden arribar a funcionar a l'hora de la veritat i ens guiarà quan hàgim d'escollir quins aplicar primer. Durant l'extracció de referències, cada vegada que s'utilitza un *wrapper* s'avalua la informació que ha extret i es corregeix la valoració sobre el funcionament d'aquest *wrapper*.

El sistema d'avaluació és molt senzill, per cadascuna de les vegades que s'extreu informació amb èxit es dóna un vot positiu. Si la informació no és correcta, se li'n dóna un de negatiu. La puntuació del *wrapper* serà el percentatge de vots positius.

$$score = \frac{vots\ positius}{vots\ totals}$$

Amb aquesta manera de calcular la puntuació, hem de ser conscients sobre què passa quan comparem *wrappers* molt utilitzats amb altres que no ho han estat tant. Per exemple, si tenim un *wrapper* que ha donat bons resultats moltes ocasions durant el passat (i.e. molts vots positius) i comença a fallar, el percentatge decreixerà i de seguida es donarà pas a un altre *wrapper* menys usat, però que no té vots negatius. Per resoldre això, moltes aplicacions usen el *Wilson score interval* descrit a [Mil09] per ordenar elements en casos semblants al nostre.

Aquest comportament que sembla problemàtic és, en certa manera, el que busquem. Quan un *wrapper* que ha funcionat gairebé sempre comença a fallar, és molt probable que ho faci a causa d'algun canvi en l'estructura de les pàgines de les quals extreu la informació. Si és així, ja sabem que aquest *wrapper* no tornarà a funcionar més i ens interessa descartar-lo tant ràpid com sigui possible. De totes maneres, la fórmula per calcular la puntuació es podria canviar fàcilment si en el futur es comencen a descartar *wrappers* bons.

Capítol 6

Anàlisi de resultats

6.1 Cerca de referències

En primer lloc provarem com de bé ho fa el sistema a l'hora de cercar pàgines a Internet que continguin informació sobre un article concret. Els tests que hem dut a terme consisteixen en els passos següents:

1. Obtenir una sèrie de consultes per cadascun dels articles d'un llistat de PDFs

2. Cercar cada consulta amb els tres cercadors implementats: *Google*, *Bing* i *Yahoo*

3. Per cada resultat obtingut, analitzem si és bo o no

4. Comptabilitzem el número de consultes que han fet falta per obtenir el primer *bon* resultat

Per tal de classificar els resultats en bons i dolents només comprovem si algunes porcions de la informació que volem es troben dins de la pàgina resultant. Aquesta no és una solució perfecta, però ens permet fer una aproximació sobre la quantitat de fitxers pels quals podem trobar la referència.

Una altra qüestió sobre la implementació d'aquestes proves, és que els resultats obtinguts se solen repetir entre consultes del mateix article. Per estalviar temps i evitar fer moltes peticions seguides als mateixos servidors (que podrien resultar en un bloqueig), deixem uns segons entre petició i petició i emmagatzemem cada resultat de manera que només l'hàgim de demanar una sola vegada. A banda d'això, en molts casos els resultats corresponen al mateix fitxer PDF del qual estem buscant informació i els hem d'ometre.

Aquests tests s'han realitzat per conjunts d'articles diferents agrupats depenent la seva capçalera, que és el que pot fer variar més els resultats obtinguts, i un últim grup amb articles de qualsevol tipus. Els gràfics de les figures 6.1 i 6.2 a mostren els resultats d'aquest últim conjunt.

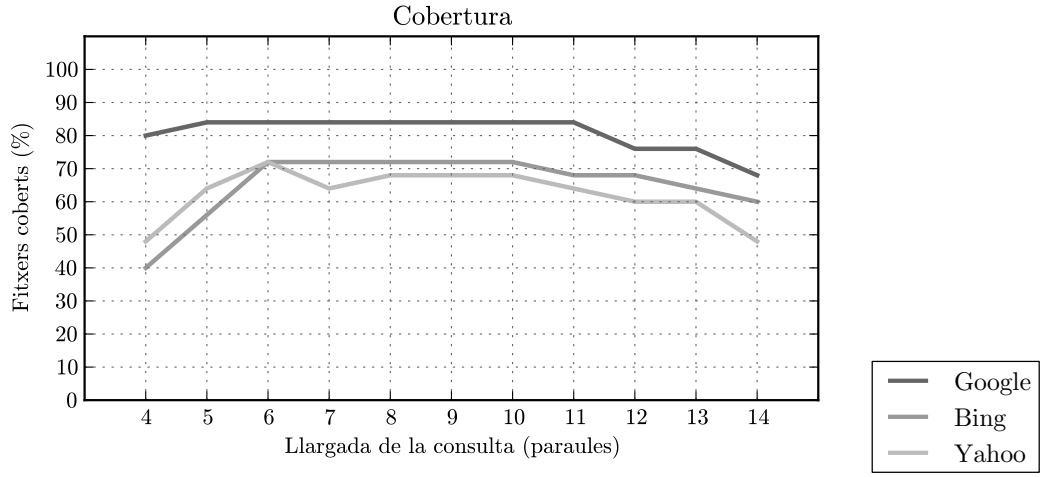


Figura 6.1: Comparació de la qualitat dels resultats obtinguts segons la llargada de les consultes

El primer gràfic mostra el percentatge de fitxers pels quals s'ha obtingut almenys un bon resultat. En algunes ocasions els resultats bons retornats per *Bing* o *Yahoo* han estat superiors en nombre, però com es pot veure, el cercador *Google* ofereix major cobertura amb resultats sobre més articles diferents.

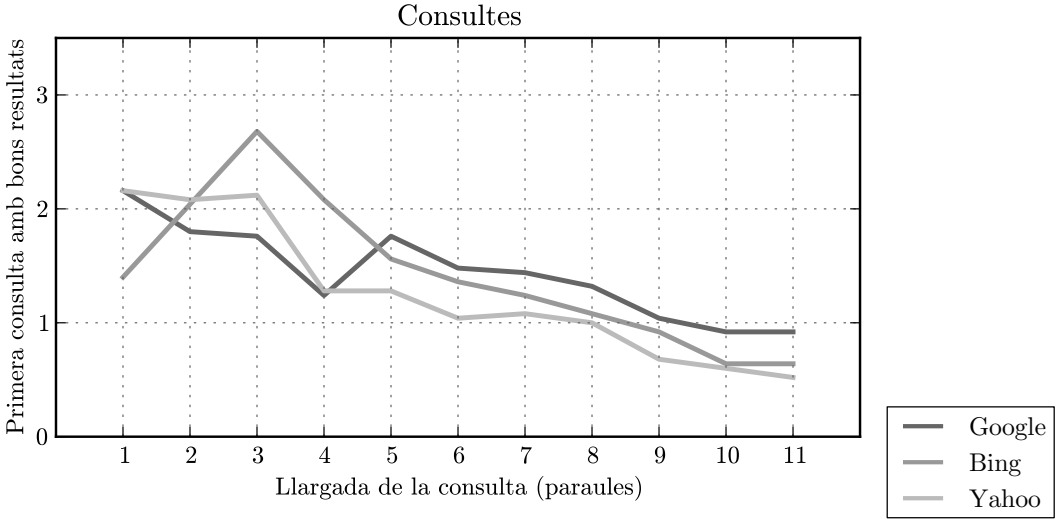


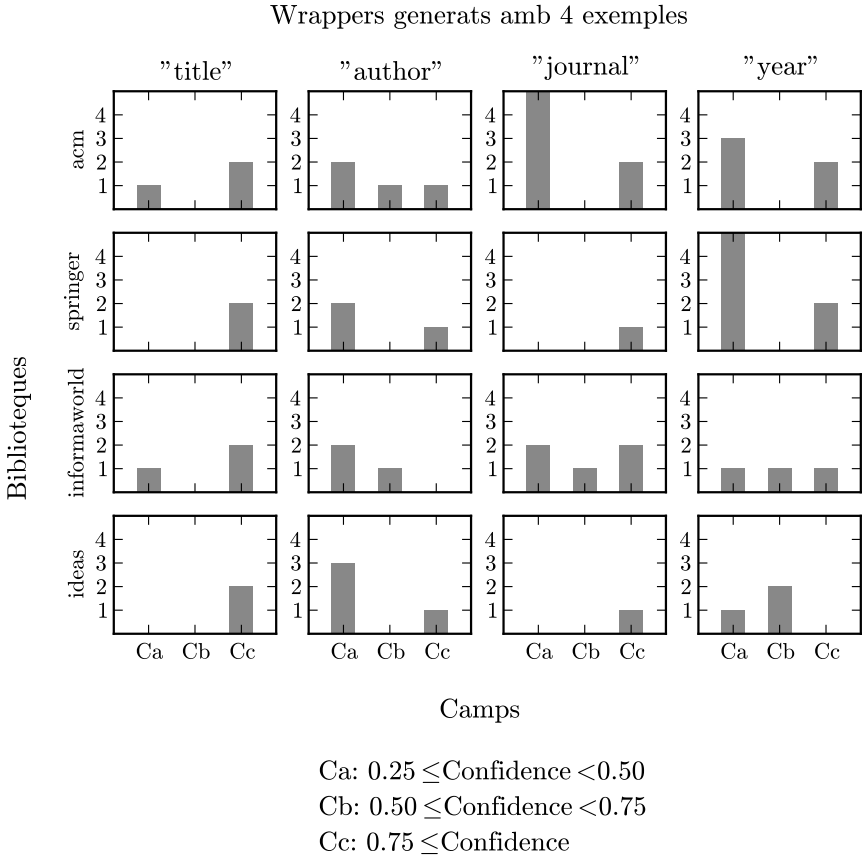
Figura 6.2: Comparació del número de consultes necessàries abans de trobar bons resultats

En el segon gràfic veiem que pel que fa a les consultes, el número de cerques que hem de fer per començar a obtenir bons resultats disminueix a mesura que es fan servir més paraules. De totes maneres, tal i com ja s'ha dit a la secció 3.2, és bo que ens saltem les primeres consultes per evitar cercar amb el títol de l'article i anar directament al resum o *abstract*.

6.2 Generació de *wrappers*

Per provar aquesta part del sistema, hem creat conjunts de pàgines web amb informació d'articles diferents i amb la referència corresponent. Cada grup inclou només pàgines corresponents a la mateixa biblioteca digital i per cadascun d'ells hem importat les referències i n'hem generat els *wrappers* pels diferents camps.

Les mostres no són gaire significatives, però ens donen una idea per poder quantificar com de bé funciona el sistema dins l'entorn pel qual està pensat. Les proves d'aquesta secció corresponen a les puntuacions rebudes durant l'avaluació dels *wrappers* i permeten marcar un ordre d'elecció inicial a l'hora d'extreure referències. Com que de moment encara no hem fet proves amb pàgines que no s'han emprat per la generació (ho farem a la propera secció), els gràfics no indiquen la correctesa dels resultats dels *wrappers* sinó la *confiança* que tenim en que funcionin. Tal i com ja s'ha dit, els camps obligatoris que han de contenir les refeerències a articles són: *author*, *title*, *journal* i *year* i són amb els que ens centrarem.



El primer gràfic permet comparar el número de *wrappers* obtinguts per cadascun dels camps per diferents biblioteques i utilitzant 4 exemples al generar. Tal i com es pot veure a la llegenda, els hem classificat en diferents grups de confiança, depenent de la puntuació rebuda a l'avaluar-los

amb aquests mateixos 4 exemples (s'han omès aquells que no han funcionat en cap cas). Aquí ens interesa veure, sobretot, si hi ha almenys un *wrappers* de confiança *Cc*. Quan n'hi més d'un, solen indicar que les dades estan repetides dins la pàgina i que, per tant, hi ha diverses formes de poder-les extreure. En el cas del camp *title* de la biblioteca *acm*, el títol es troba a l'etiqueta `<title>` i dins d'alguna altra etiqueta dins el `<body>` de la pàgina.

El gràfic també mostra dos casos pels quals no s'ha obtingut cap *wrapper* que hagi funcionat per tots els exemples: el camp *author* de la biblioteca *InformaWorld* i el camp *year* d'*Ideas*. Amb una ullada ràpida a les regles generades n'hi ha prou per veure que es tracta de problemes de generalització. Pel cas del camp corresponent a l'any, les expressions regulars de les dues *regex rules* són:

```
"Handle\:\ RePEc\:tov\:dsiess\:v\:3\:y\:(.*)"
```

```
"Handle\:\ RePEc\:(?:.*)af\:(?:.*)v\:(?:.*)\:y\:(.*)"
```

Aquest és el problema que s'ha descrit a la secció 5.2.2 al parlar de la generació d'expressions regulars. Els patrons inicials eren massa diferents com per fusionar-los i s'ha acabat amb regles massa específiques. Si ho corregim manualment, l'expressió resultant seria:

```
"Handle\:\ RePEc\:(?:.*)\:y\:(.*)"
```

Aquí queda plasmada ara la importància d'escollir bons exemples per la generació de regles. A l'apèndix B.4 hi ha la mateixa gràfica, però corresponent als *wrappers* obtinguts utilitzant només 2 exemples. Els resultats són similars.

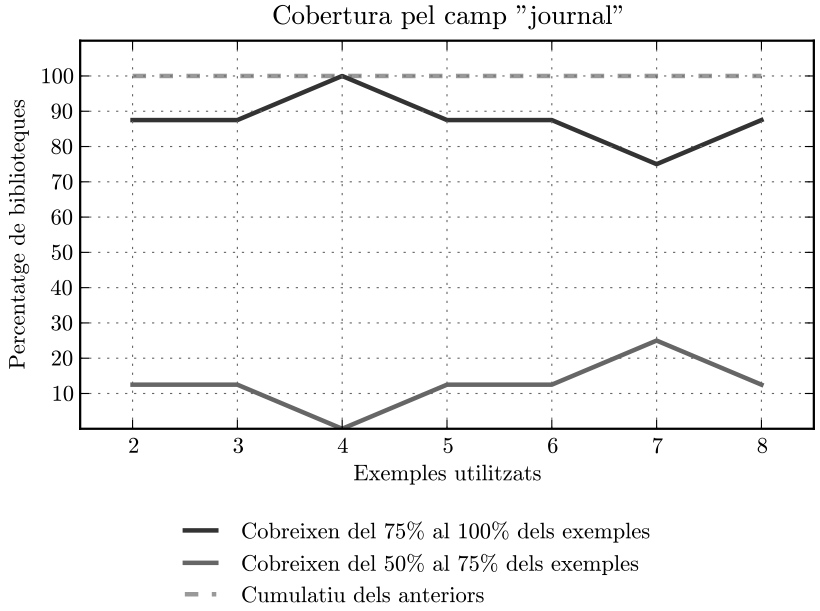


Figura 6.3: Cobertura dels *wrappers* pel camp *journal*

A part d'aquesta vista dels resultats més detallada, les gràfiques 6.3 i 6.4 també mostren el percentatge de biblioteques provades per les quals s'han obtingut *wrappers* de confiança, depenent del número d'exemples utilitzats per la generació. La línia fosca correspon al percentatge de les biblioteques digitals per les quals hem obtingut almenys un *wrapper* de confiança màxima, la línia més clara indica el mateix percentatge, però corresponent al següent interval de confiança. Finalment, la línia discontinua representa la suma de les dues anteriors.

El camp corresponent a l'any és interessant perquè moltes de les pàgines que hem provat contenen múltiples aparicions del valor que busquem, però no sempre descrivint l'any de publicació, sinó la data de revisió, la data a partir de la qual l'article es troba a Internet, *copyright*, etc. Aquest tipus de confusions també són habituals quan, a més de la informació de l'article, les pàgines inclouen llistats amb les citacions o referències a d'altres publicacions. No obstant, a mesura que comencem a tenir més exemples l'avaluació ens permetrà escollir aquells que realment són vàlids.

Altres camps pels quals es té més dificultat per generar *wrappers* són aquells el valor dels quals consisteix en números petits, com ara el número de volum (`volume`) o de revista (`number`). El motiu és el mateix, hi ha moltes aparicions del mateix valor, però que no fan referència al camp que busquem.

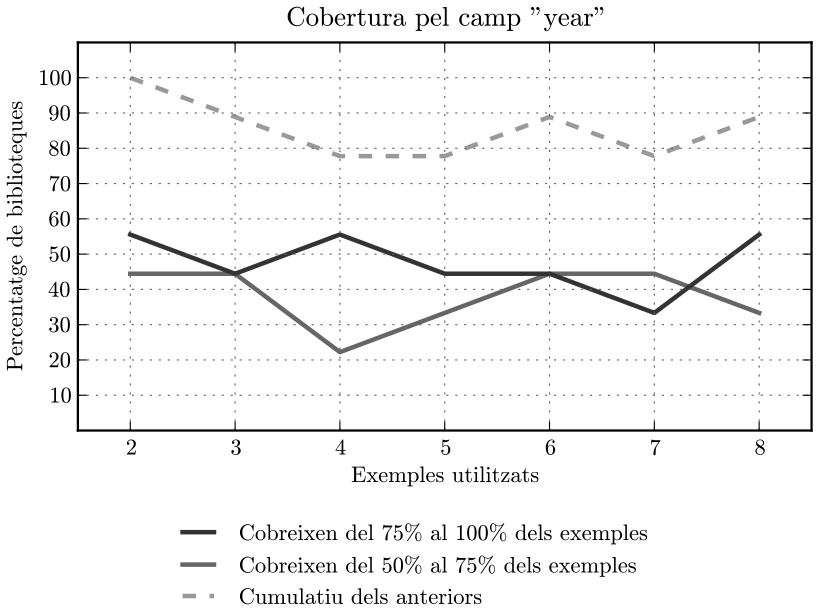
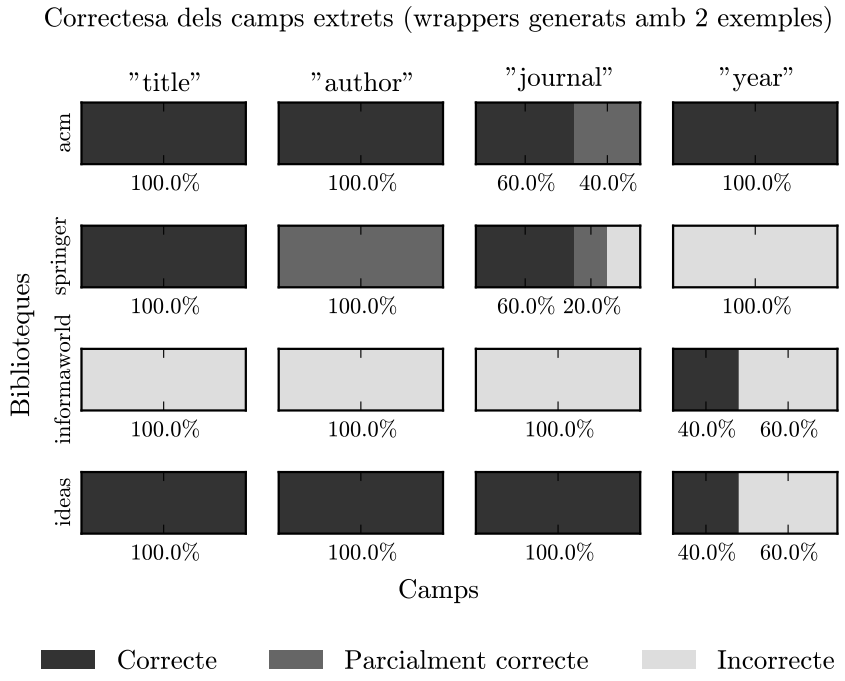


Figura 6.4: Cobertura dels *wrappers* pel camp *year*

6.3 Extracció de referències

Anem a veure ara com de bé ho fan els *wrappers* generats a la secció anterior a l'hora d'extreure informació per les mateixes biblioteques i camps. Disposem de conjunts de pàgines corresponents

a articles diferents i les referències en BibTeX que faran de mostres de control per saber en quins casos s'ha encertat i en quins no. Com que en aquest punt els resultats són més interessants, hem inclòs els gràfics de 2 i 4 exemples aquí mateix. Per començar ens fixarem en la correctesa dels camps extrets amb regles generades a partir de dos exemples:



Es considera que el valor extret és:

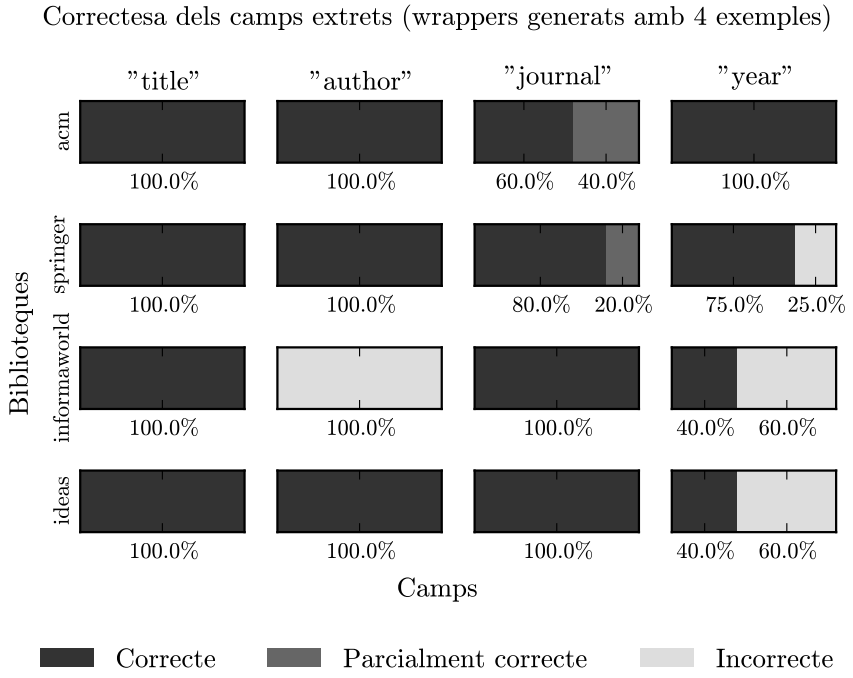
- *Correcte*: Quan el valor obtingut coincideix exactament amb el de la referència de control.
- *Parcialment correcte*: Si el valor extret conté el valor de control a més d'altra informació. Per exemple, un dels valors extrets pel camp *journal* de la biblioteca *ACM* és: `ACM Computing Surveys (CSUR)` mentre que el valor de la referència de control és `ACM Computing Surveys`. Tot i que en situacions com aquestes els valors es podrien considerar correctes, s'ha decidit generar els gràfics aplicant les regles de classificació de forma estricta i comentar-ho si fa falta.
- *Incorrecte*: Qualsevol altre cas.

Veiem que utilitzant dos exemples hi ha hagut força problemes, tots per culpa que les regles són massa específiques com per cobrir altres pàgines:

- Com ja s'ha anticipat a la secció anterior, un dels camps amb més problemes ha estat l'any, que no s'ha pogut extreure correctament en tres de les quatre biblioteques que es mostren per culpa que no s'ha escollit bé l'element HTML que realment conté aquesta informació.
- Els autors de la biblioteca *SpringlerLink* no s'han extret del tot bé en cap dels casos, per culpa que els separadors de les *separator rules* no són prou genèrics.

- Per últim, els resultats de la biblioteca *InformaWorld* són pèssims a causa de combinacions dels problemes anteriors.

Anem a veure què passa quan fem les mateixes proves amb els *wrappers* generats a partir de quatre exemples diferents:



El nombre d'extraccions correctes ha augmentat força, i els valors marcats com a *parcialment correctes* corresponen a situacions que realment es podrien considerar vàlides. En canvi, segueixen havent-hi problemes amb els anys i els autors de la biblioteca *InformaWorld*. Podem mirar de corregir-los manualment.

Respecte als autors d'*InformaWorld* no s'ha sabut identificar correctament l'element HTML que els conté. Només canviant l'expressió regular de l'inici de la *path rule*, per ajudar a escollir l'element adequat, passem a extreure els autors de forma correcta per a tots els casos.

```
[".*", ["div", {"id": "metahead"}, 0], ...
```

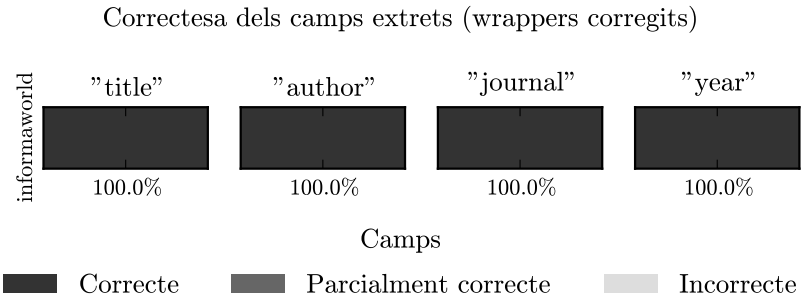
```
["Authors?:\ (.*)", ["div", {"id": "metahead"}, 0], ...
```

Pel que fa a l'any de publicació, el problema consisteix, una vegada més, en una expressió regular massa específica:

```
"Published\ in\:Accounting\ Education\,\ Volume\ \ 1(?:.*)
    \,\ Issue\ \ (?:.*)\ \&\ (?:.*)\ \ (?:.*)\ (.*)\ \,\
    pages\ (?:.*)\ \-\ "
```

```
"\ (\d{4})\ \,\ pages\ "
```

Un cop fetes aquestes correccions, si tornem a executar les proves per extreure les referències, obtenim un 100% d'encerts pels quatre camps que hem tractat.



Els resultats d'extracció dels camps que no es mostren als gràfics (e.g. número de volum, pàgines) són molt similars als que acabem de veure. El número de revista i volum tenen problemes similars als del camp referent a l'any, en canvi, altres menys comuns com ara l'ISSN una taxa d'encert molt alta.

Capítol 7

Conclusions i treball futur

Al llarg d'aquest document s'han descrit les decisions que hem pres i el procés seguit per poder implementar una aplicació d'extracció de referències. Partíem d'aquest objectiu únic i, per tant, tot el que hem anat fent gira al voltant d'aquesta idea: l'extracció de text dels PDFs, la cerca d'articles en biblioteques digitals i l'extracció d'informació dins d'un document HTML.

El procés d'extracció està basat en un cúmul d'assumpcions que, dins del context de l'aplicació, acostumen a complir-se en la majoria de casos. De totes maneres, cal tenir en compte que el procés es veu afectat per pèrdues que són inevitables a cada pas:

- L'extracció del text dels fitxers PDF no es pot fer per documents formats per imatges que han estat escanejats sense utilitzar *software* de reconeixement de caràcters.
- El pas de cercar el document a Internet requereix, com és lògic, que es trobi en alguna de les biblioteques digitals indexades pels cercadors.
- Per últim, pel que fa a extreure informació de les pàgines HTML de forma correcta, ja hem vist que depèn molt de les regles que tinguem definides, però que els resultats tampoc no són perfectes.

Al final, hem aconseguit una eina que funciona i que, tal i com hem pogut comprovar, obté resultats mitjanament bons. L'èxit de regles generades per a l'extracció varia segons el tipus de dades a obtenir. Mentre els camps com ara el títol, autors, nom de revista, etc. tenen una taxa d'encerts força elevada, altres camps com l'any de publicació, número de volum o número de la revista porta més problemes.

Cal comentar que, tot i que el sistema està destinat a l'extracció de referències, la part de generació de regles i la manera d'aplicar-les podria ser, en realitat, la base d'un *framework* útil per qualsevol altre sistema similar d'extracció d'informació estructurada. En cas de no tractar-se de documents HTML, només caldria afegir els nous tipus de regles necessaris.

Un altre punt interessant és que, al tractar-se d'un projecte *open source*, qualsevol persona interessada es pot dirigir al repositori públic que hi ha a *GitHub*[Rep] i fer un *fork* per continuar desenvolupant. Això sí, qualsevol treball derivat haurà de distribuir-se sota les mateixes condicions.

7.1 Possibles Millores

Des d'un punt de vista objectiu, el sistema encara hauria de passar per més iteracions abans de poder-lo considerar sòlid. Alguns dels aspectes que s'haurien de tractar primer són:

- El primer i més important seria incorporar millores a la generació d'expressió regulars a partir d'exemples per tal d'aconseguir expressions generals utilitzant pocs exemples.
- Possibilitat d'actualitzar *wrappers* a mesura que es disposa de nous exemples. Quan una referència s'extreu correctament de forma automàtica significa que les regles funcionen, però quan hi ha errors i l'usuari fa correccions, l'aplicació hauria de ser capaç d'adonar-sen i actualitzar-se per fer-ho bé en pròximes execucions. Hauria de ser capaç d'aprendre.
- Suportar altres llenguatges de citacions no només suposaria poder importar i exportar referències amb aquests formats sinó que permetria que els *reference wrappers* que hem descrit a la secció 4.1.1 també puguessin extreure aquest tipus referències.
- Millorar la interfície d'usuari; afegint funcionalitats, però també en termes d'usabilitat. Per exemple: caldria arreglar-la de manera que hi hagi consistència en la manera de realitzar les mateixes operacions per totes les parts de l'aplicació.
- També referent a la interfície, seria convenient tenir opcions per poder configurar tots els paràmetres de l'aplicació que actualment s'estableixen dins del fitxer de configuració.

Veiem, doncs, que encara queda molt per fer, tant com vulguem.

Bibliografia

[GBL98] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: an automatic citation indexing system. In *International Conference on Digital Libraries*, pages 89–98. ACM Press, 1998.

[GPL] Gnu general public license. `http://www.gnu.org/licenses/gpl.html`.

[Mil09] Evan Miller. How not to sort by average rating. `http://www.evanmiller.org/how-not-to-sort-by-average-rating.html`, 2009.

[Pyta] Difflib - helpers for computing deltas. `http://docs.python.org/library/difflib.html`.

[Pytb] Easy install. `http://peak.telecommunity.com/DevCenter/EasyInstall`.

[Pytc] Regular expression operations in python. `http://docs.python.org/library/re.html`.

[Rep] Bibtex index maker al github. `http://github.com/rxuriguera/bibtexIndexMaker`.

[RM88] John Ratcliff, W. and David Metzener. Pattern matching: The gestalt approach. *Dr. Dobb's Journal*, page 46, 1988.

[Sod99] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272, 1999.

Apèndix A

Biblioteques utilitzades

Per una banda fem servir els següents mòduls de la biblioteca estàndard; no són els únics que utilitzem, però sí els que ofereixen funcionalitats més interessants. De fet, *Python* té una biblioteca estàndard molt extensa per la qual cosa és corrent descriure-la amb les paraules *batteries included*.

- RE: Ofereix la possibilitat de treballar amb expressions regulars. Com ja hem vist, les expressions regulars han han tingut molta importància i les hem fet servir àmpliament a tot el sistema.

- SimpleJSON: L'hem utilitzat per dos objectius diferents: interactuar amb les APIs dels cercadors i per serialitzar i desserialitzar objectes *Python* a l'emmagatzemar-los a la base de dades. Els objectes serialitzats utilitzant aquest mòdul tenen la característica que són llegibles per les persones. Per exemple, el resultat de serialitzar un diccionari és: `'{``clau01'': 4, ``clau03'': 15, ``clau02'': 8}'`. Això permet que sigui fàcil editar-los manualment.

- DiffLib: Ens permet fer *string matching* i obtenir llistes de blocs coincidents així com un índex de similaritat entre dues cadenes de caràcters. L'utilitzem a l'hora de generar regles automàticament, per tal de decidir quan hem de fusionar regles i com ho hem de fer.

- ConfigParser: Mòdul que ens permet llegir el fitxer de configuració de l'aplicació al qual hi ha definits els diferents paràmetres que guien l'aplicació.

També fem ús de les biblioteques següent:

- XGoogle: Es tracta d'una biblioteca força senzilla escrita per Peteris Krumins que facilita les cerques a Internet. L'objectiu del creador era oferir la possibilitat de cercar a *Google* evitant les limitacions de la API oficial, que només deixa obtenir un màxim de 32 resultats. Nosaltres l'hem ampliat per poder obtenir resultats de *Google Scholar* i posteriorment, també per utilitzar les APIs oficials JSON de *Google*, *Bing* i *Yahoo*.

- BeautifulSoup: Es tracta d'un *parser* d'HTML i XML desenvolupat per Leonard Richardson que genera arbres sintàctics sobre els quals podem operar. Una de les funcionalitats que més fem servir és la de cercar elements segons les etiquetes HTML, els atributs, o bé el text que contenen.

- *Parser* de BibTeX: Es tracta d'un *parser* escrit per Raphael Ritz pel projecte Bibliograph.parsing. El fem servir per obtenir els diferents camps de les referències a l'hora d'importar-les a l'aplicació i a l'extreure-les mitjançant *reference wrappers* (veure 4.1.1) per tal de poder-les validar.
- *SQLAlchemy*: És un ORM per *Python* que falicita el treball amb la base de dades. A més, ofereix una capa d'abstracció que permet utilitzar la base de dades de forma independent a la tecnologia. Per exemple, nosaltres fem servir *SQLite*, però en teoria podríem fer un canvi del sistema gestor de la base de dades sense haver de tocar res més.
- PyQt4: Es tracta d'una biblioteca en *Python* que embolcalla el *framework* per la creació d'interfícies gràfiques *Qt* 4.

Apèndix B

Resultats dels tests

A continuació es mostren els gràfics amb la resta de resultats de les proves que s'han realitzat. La majoria corresponen al mateix tipus de tests que els del capítol 6, però variant algun paràmetre.

B.1 Cerca de referències

Les gràfiques següents són semblants a la de la figura 6.1. En aquest cas, però, el tipus de fitxers per als quals extraiem consultes i fem les cerques estan agrupats en dues categories segons l'estructura del seu contingut. Per un costat, a la figura B.1, tenim fitxers que tenen una pàgina sencera com a capçalera abans del resum o *abstract* de l'article.

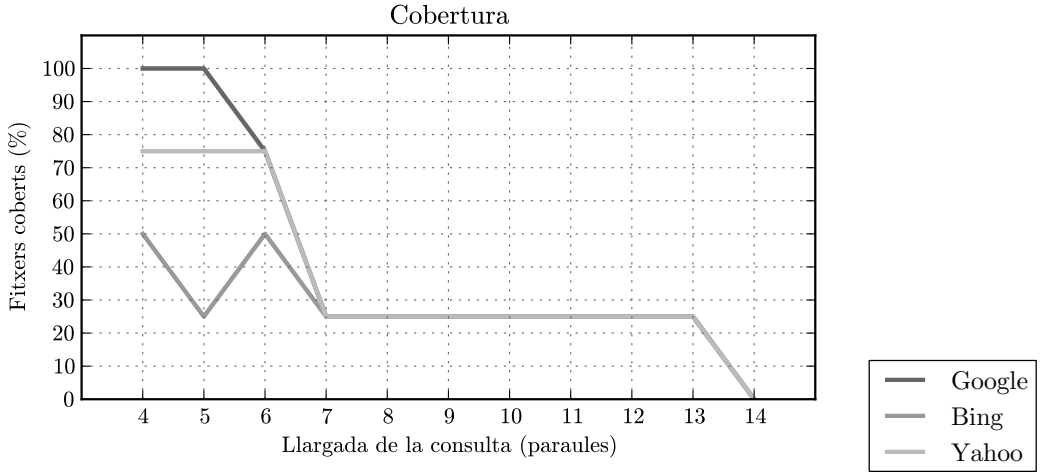


Figura B.1: Qualitat dels resultats per fitxers amb una pàgina sencera com a capçalera

Veiem que per aquest tipus d'article, el percentatge de pàgines per les quals s'han obtingut bons resultats disminueix molt a mesura que s'augmenta la llargada de la consulta. Cal tenir en compte que les consultes s'han obtingut totes a partir de la primera pàgina del fitxer. Al no contenir el resum, fa que l'expressió regular usada per trobar la consulta no tingui coincidències

per la majoria dels articles provats. Si executem les mateixes proves, però agafant les consultes de les dues primeres pàgines, els resultats milloren força:

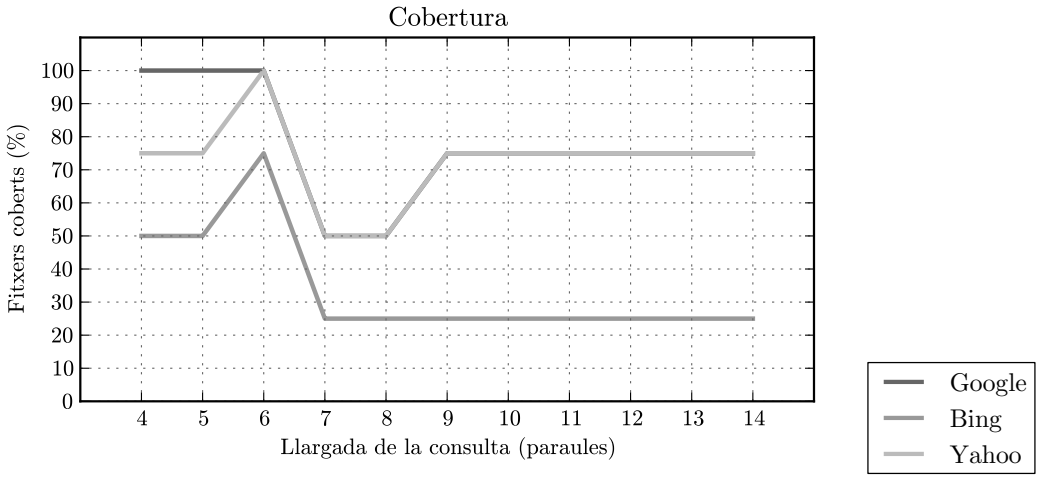


Figura B.2: Qualitat dels resultats per fitxers amb una pàgina sencera com a capçalera II

El gràfic de la figura següent s'ha obtingut a partir d'articles que tenen una capçalera *normal*. Considerem que les capçaleres dels articles més habituals són aquelles que tenen l'*abstract* a la mateixa pàgina.

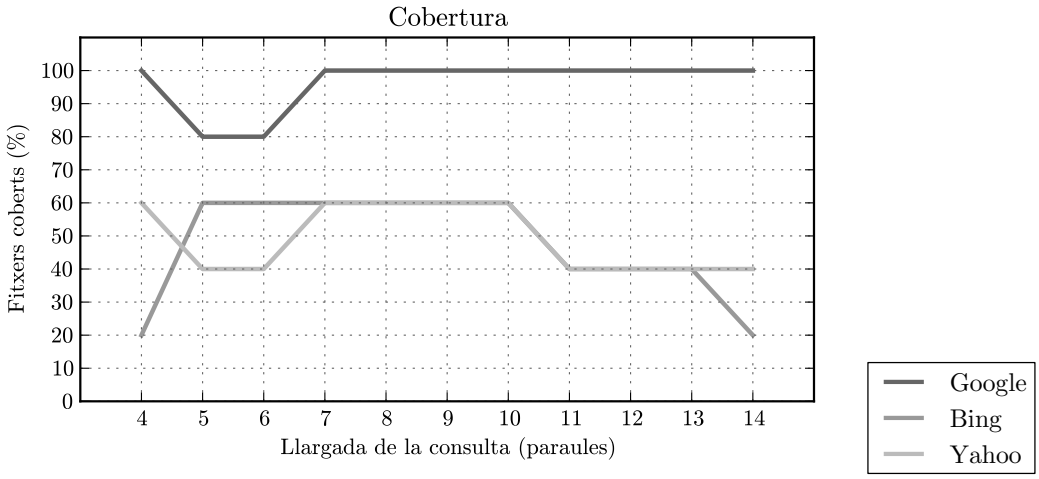


Figura B.3: Qualitat dels resultats per fitxers amb capçaleres *normals*

Pel que fa al número de consultes necessàries per començar a obtenir bons resultats, els gràfics per aquests dos conjunts de fitxers són molt semblants al que hem vist a la figura 6.2.

B.2 Generació de *wrappers*

Pel que fa a la generació automàtica de regles d'extracció, la figura següent mostra els *wrappers* obtinguts al generar-los fent servir només dos exemples. Com es pot comprovar, no se n'han generat tants com a l'utilitzar quatre exempes, però també hi ha *wrappers* de confiança màxima per a la majoria de camps. Una altra cosa és que realment funcionin a l'executar-los per nous documents, tal i com hem vist que passava.

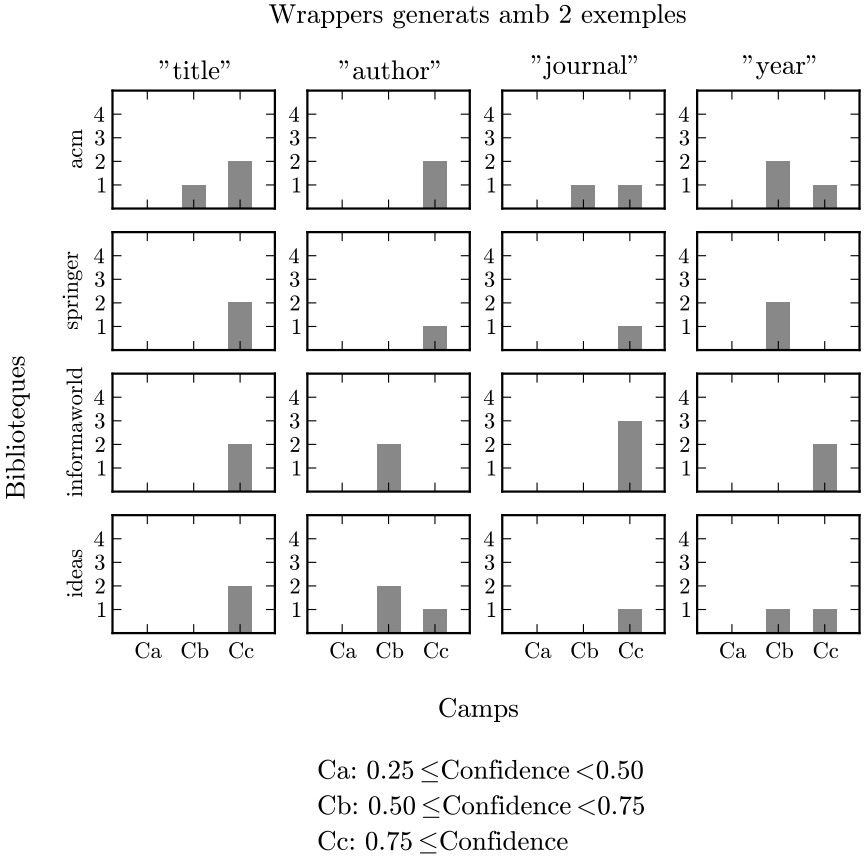


Figura B.4: Nombre de *Wrappers* generats utilitzant 2 exemples i agrupats per confiança

També hem mirat la *cobertura* dels *wrappers* generats per altres camps a part del nom de la revista i l'any. Els gràfics de la figura B.5 en mostren els resultats. És interessant comparar la confiança dels *wrappers* d'autors, un camp multi-valor relativament difícil d'exterue; i la de les pàgines i el número de volum. Els motius són els que ja s'han comentat a la secció de resultats.

Per últim, també hem generat un gràfic (B.6) que reflecteix una mitjana del temps necessari per generar els *wrappers* per un camp qualsevol depenent del número d'exemples que s'utilitzen.

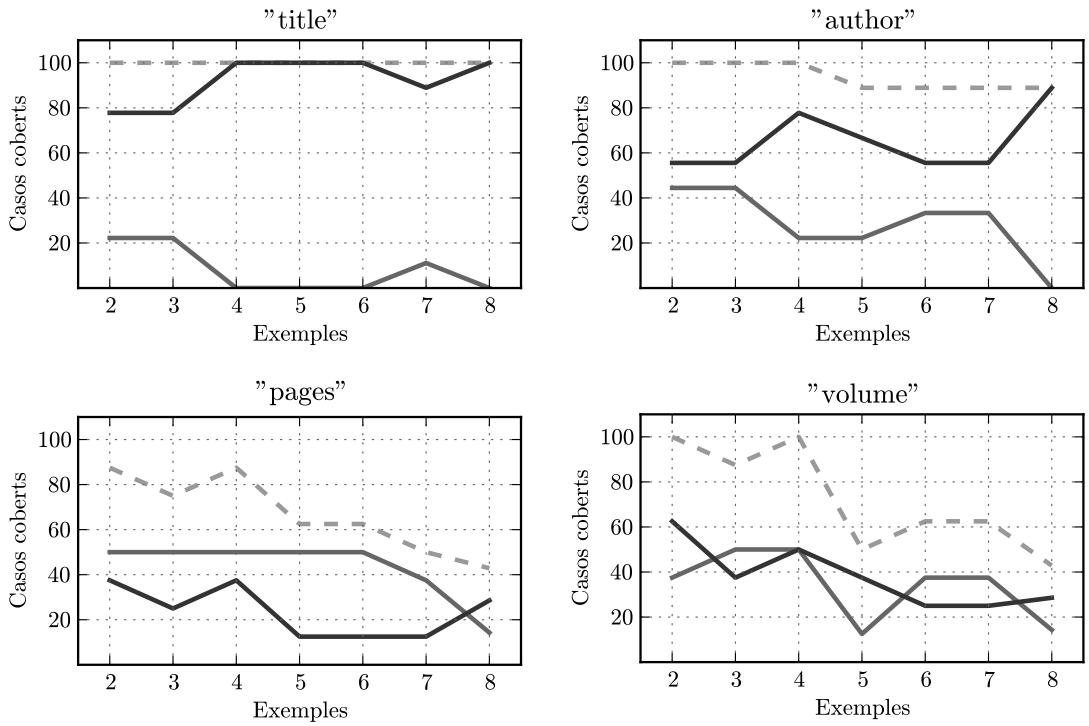


Figura B.5: Cobertura dels *wrappers* generats

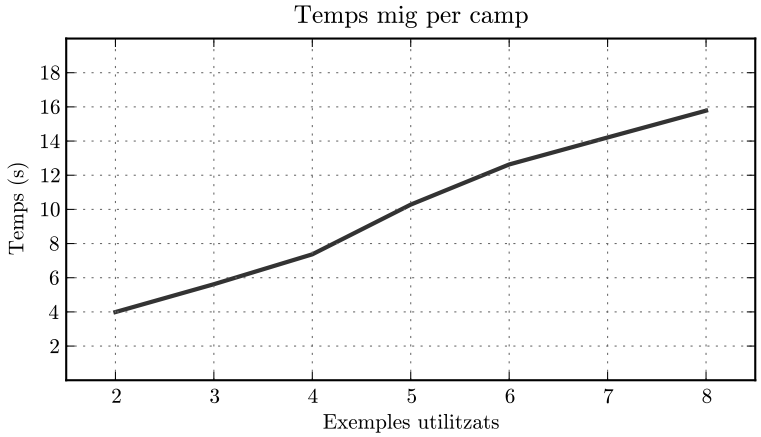


Figura B.6: Temps mitjà per la generació dels *wrappers* d'un únic camp

Apèndix C

Diagrames

En aquest apèndix es mostren alguns diagrames que poden ajudar a entendre una mica millor l'aplicació. Creiem que cada diagrama és autoexplicatiu només s'acompanyen d'una petita descripció als peus de cadascun d'ells.

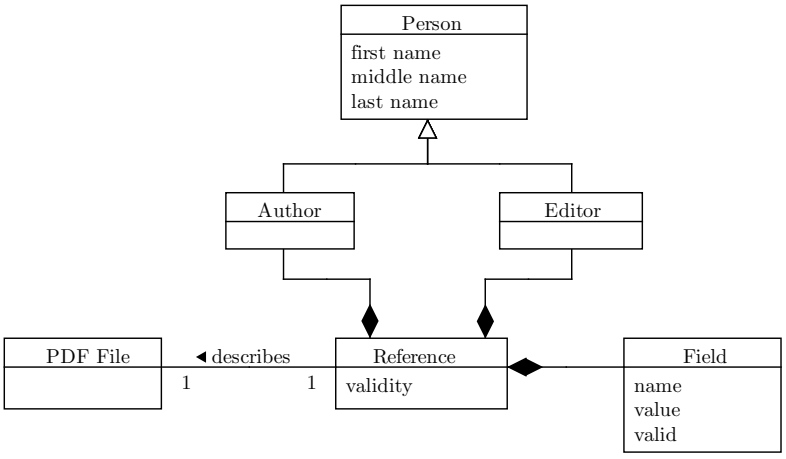


Figura C.1: Model de domini respecte a referències

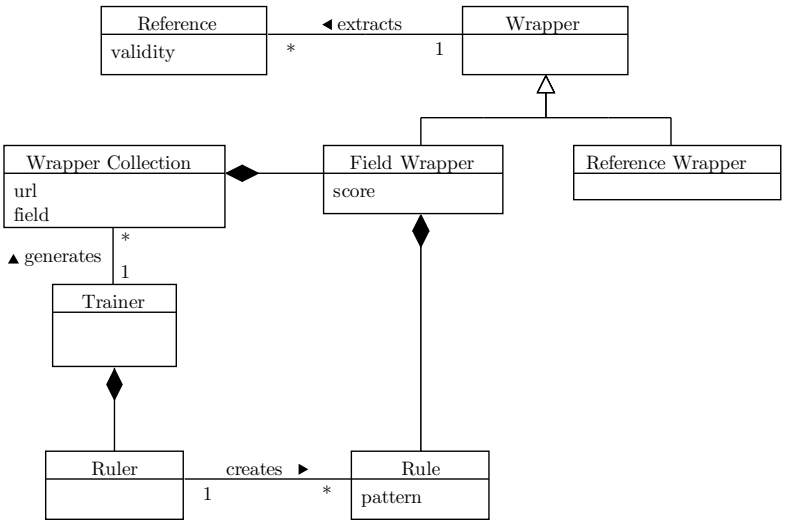


Figura C.2: Model de domini dels *wrappers*

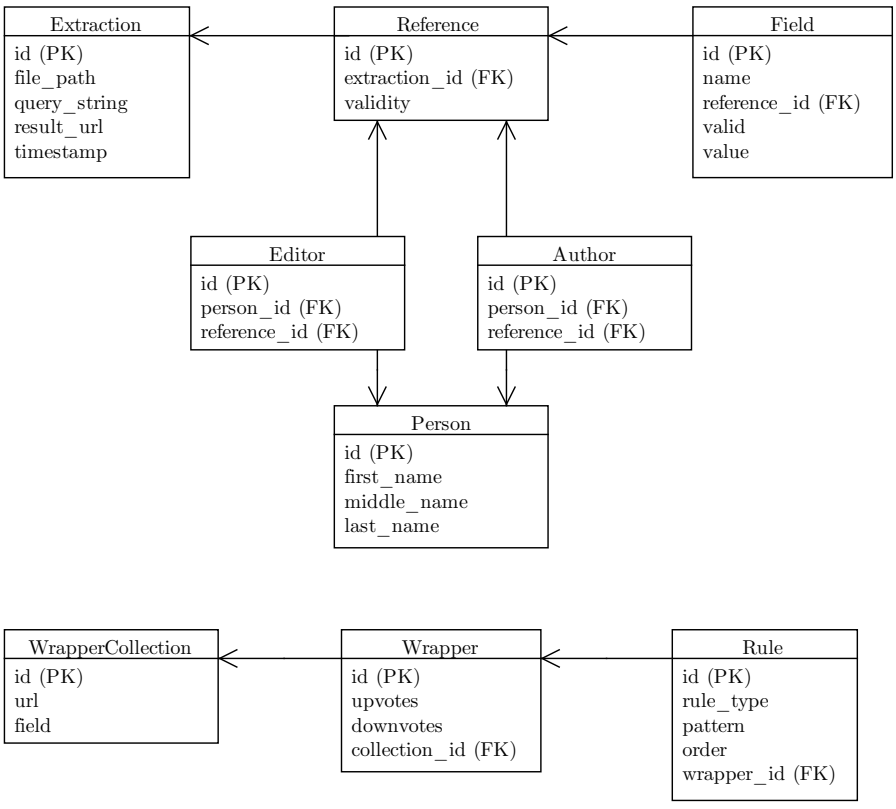


Figura C.3: Estructura de la base de dades

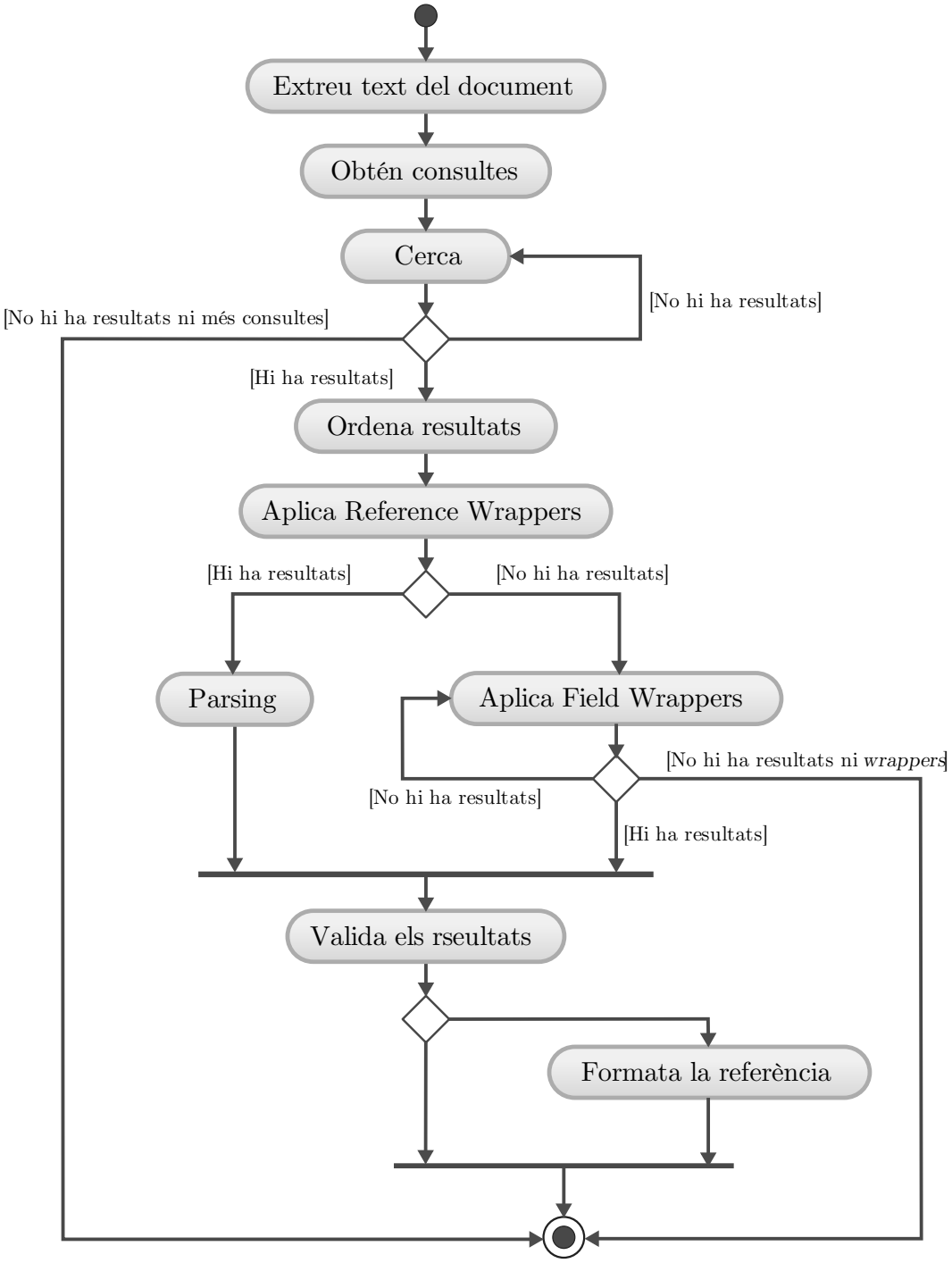


Figura C.4: Diagrama d'activitat per a l'extracció de referències

Apèndix D

Extracció Contingut PDF

A continuació es mostren exemples de com queden les capçaleres d'alguns fitxers PDF a l'extreure-les en forma de text.

D.1 Exemple 01

Estructura del PDF

ELSEVIER

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Discrete Applied Mathematics 147 (2005) 43–55

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Pseudo-models and propositional Horn inference

Bernhard Ganter*, Rüdiger Krauße

Institut für Algebra, Technische Universität Dresden, Zellescher Weg 12-14, D-01062 Dresden, Germany

Received 4 May 2001; received in revised form 6 January 2003; accepted 21 June 2004
Available online 29 December 2004

Abstract

A well-known result is that the inference problem for propositional Horn formulae can be solved in linear time. We show that this remains true even in the presence of arbitrary (static) propositional background knowledge. Our main tool is the notion of a cumulated clause, a slight generalization of the usual clauses in Propositional Logic. We show that each propositional theory has a canonical irredundant base of cumulated clauses, and present an algorithm to compute this base.

Text

Discrete Applied Mathematics 147 (2005) 43 55 www.elsevier.com/locate/dam

Pseudo-models and propositional Horn inference Bernhard Ganter , Rüdiger Kraußе Institut für Algebra, Technische Universität Dresden, Zellescher Weg 12-14, D-01062 Dresden, Germany Received 4 May 2001; received in revised form 6 January 2003; accepted 21 June 2004 Available online 29 December 2004

Abstract A well-known result is that the inference problem for propositional Horn formulae can be solved in linear time. We show that this remains true even in the presence of arbitrary (static) propositional background knowledge. Our main tool is the notion of a cumulated clause, a slight generalization of the usual clauses in Propositional Logic. We show that each propositional theory has a canonical irredundant base of cumulated clauses, and present an algorithm to compute this base. © 2004 Elsevier B.V. All rights reserved. MSC: 03B05; 03B35; 68T30 Keywords: Horn inference; Horn base; Background knowledge

D.2 Exemple 02

Estructura del PDF

Closed and Maximal Tree Mining Using Natural Representations

José L. Balcázar balqui@lsi.upc.edu
Albert Bifet abifet@lsi.upc.edu
Antoni Lozano antoni@lsi.upc.edu
Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics

1. Introduction

Mining frequent trees is becoming an important task, with broad applications including chemical informatics, computer vision, text retrieval, bioinformatics, and Web analysis. Many link-based structures may be studied formally by means of unordered trees [...]

the closed graph patterns discovered.

In the case of trees, there are two broad kinds of subtrees considered in the literature: subtrees which are just induced subgraphs, called induced subtrees, and subtrees where contraction of edges is allowed, called embedded subtrees[...]

Text

Closed and Maximal Tree Mining Using Natural Representations

Jos' L. Balc'zar e a balqui@lsi.upc.edu Albert Bifet abifet@lsi.upc.edu Antoni Lozano antoni@lsi.upc.edu Universitat Polit'cnica de Catalunya, Departament de Llenguatges i Sistemes Inform'tics e a

1. Introduction Mining frequent trees is becoming an important task, with broad applications including chemical informatics, computer vision, text retrieval, bioinformatics, and Web analysis. Many link-based structures may be studied formally by means of unordered trees.

D.3 Exemple 03

Estructura del PDF

Applied Economics, 2010, 42, 825–850

Routledge
Taylor & Francis Group

Modelling the interactions across international stock, bond and foreign exchange markets

Abdul Hakim[a,*] and Michael McAleer[b]

[a]Department of Economics, University of Western Australia, Australia and Faculty of Economics, Indonesian Islamic University, Indonesia
[b]Department of Economics, University of Western Australia, Australia

The benefits of investing internationally depend on three conditions, namely, cross-country correlations, market volatilities and future changes in currency risks (Odier and Solnik, 1993). This article investigates these conditions for several countries. Many papers have modelled both domestic interactions across asset markets and international interactions in individual asset markets in isolation, but rarely have they examined international interactions across asset markets. The article fills this gap by modelling the international interactions across stock, bond and foreign exchange markets. Two models that meet these purposes are the VARMA-AGARCH model of Hoti et al. (2002) and the VARMA-GARCH model of Ling and McAleer (2003). The countries that will be modelled in this article are Australia, Japan, Singapore, New Zealand and USA.

Text

Applied Economics, 2010, 42, 825850

Modelling the interactions across international stock, bond and foreign exchange markets
Abdul Hakima,* and Michael McAleerb
Department of Economics, University of Western Australia, Australia and Faculty of Economics, Indonesian Islamic University, Indonesia b Department of Economics, University of Western Australia, Australia a

Downloaded By: [Consorci de Biblioteques Universitaries de Catalunya] At: 11:49 20 May 2010

The benefits of investing internationally depend on three conditions, namely, cross-country correlations, market volatilities and future changes in currency risks (Odier and Solnik, 1993). This article investigates these conditions for several countries. Many papers have modelled [...]

Apèndix E

Manual d'usuari

L'aplicació es distribueix en forma d'`egg` de *Python* i pot ser instal·lada de la manera estàndard amb la comanda `easy_install` (més detalls a [Pytb]). Aquest capítol se centra en descriure alguns dels aspectes d'utilització de l'aplicació un cop instal·lada i mitjançant la interfície gràfica. Bàsicament, tindrem un amb dues grans categories: *references* i *wrappers*. Vegem què hi ha en cadascuna d'elles.

E.1 Referències

Extracció

La primera opció del menú és la que permet llegir un directori de fitxers PDF i obtenir-ne les referències. El sistema inclou *wrappers* per les biblioteques digitals que s'han descrit en altres capítols d'aquest document, i per tant, des d'un principi ja es pot provar a fer extraccions d'articles indexats per aquestes biblioteques.

La vista de la figura E.1 mostra el camp únic que cal omplir per tal de començar l'extracció: simplement seleccionem un directori o fitxer i cliquem el botó *Extract References*. Durant l'execució, es mostra una barra de progrés i una vegada finalitzat, se'ns mostrarà una finestra molt semblant a la de la figura E.4 indicant-nos que podem veure i editar les referènceies amb l'opció del menú *Manage*.

Gestió

Amb l'opció *Manage* (veure E.2) podrem gestionar les referències disponibles i realitzar les operacions CRUD típiques. A la part esquerra es mostra un llistat de totes les referències de la base de dades. Al seleccionar-ne una s'omplirà l'editor de la meitat dreta de la finestra amb tota la informació sobre la referència.
Algunes característiques:

- Podem crear nous camps, autors i editors simplement afegint-los al final de la llista adequada, a la línia buida que hi ha. Per eliminar-los, només cal esborrar totes les columnes i deixar la fila en blanc.
- Per afegir o eliminar noves referències podem fer-ho a partir del menú contextual que apareix al clicar amb el botó secundari sobre la llista *Available References*.

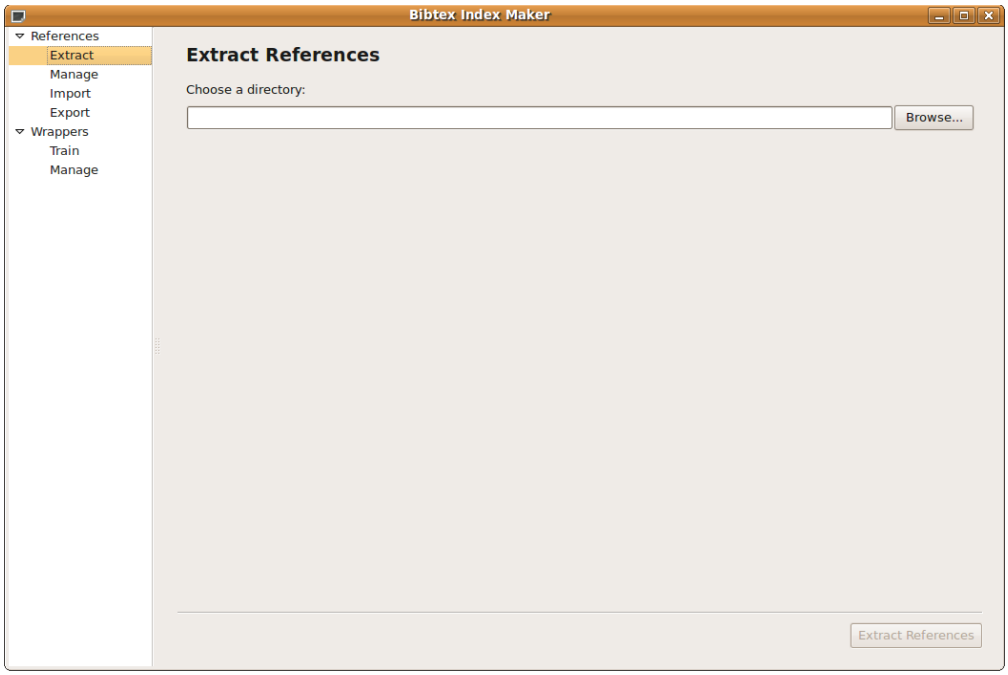


Figura E.1: Extreu referències

- És necessari definir una URL com a valor del camp *Extracted from* si posteriorment es pretén utilitzar la referència en qüestió per generar *wrappers*.
- Pel que fa al camp *File Path*, només serveix per ajudar a l'usuari a distingir entre les múltiples referències disponibles de la llista *Available References*.

Importació

Aquesta opció és la que probablement més ens interessarà la primera vegada que executem l'aplicació just després d'instal·larla. Permet importar referències des de fitxers `.bib` de manera que sigui ben fàcil començar a generar *wrappers* i configurar el sistema segons les necessitats de cadascú. El funcionament és molt semblant al de les extraccions. Se selecciona un fitxer a la finestra E.3 i un cop s'ha acabat, se'ns redirigirà a la vista de la figura E.4.

Exportació

L'última opció que queda relacionada amb les referències és la funcionalitat que permet exportar-les en format BibTeX. De la mateixa manera que amb la finestra *Manage*, a la part esquerra se'ns lllistaran totes les referències disponibles. Podrem seleccionar les que vulguem i la seva entrada formatada apareixerà a l'àrea de text de la part dreta. Hi ha la possibilitat de seleccionar o bé desseleccionar tots els elements de la llista a partir del menú contextual que apareix al fer clic amb el botó secundari. També podem desar totes les entrades seleccionades directament a un fitxer amb el botó *Save to file*.

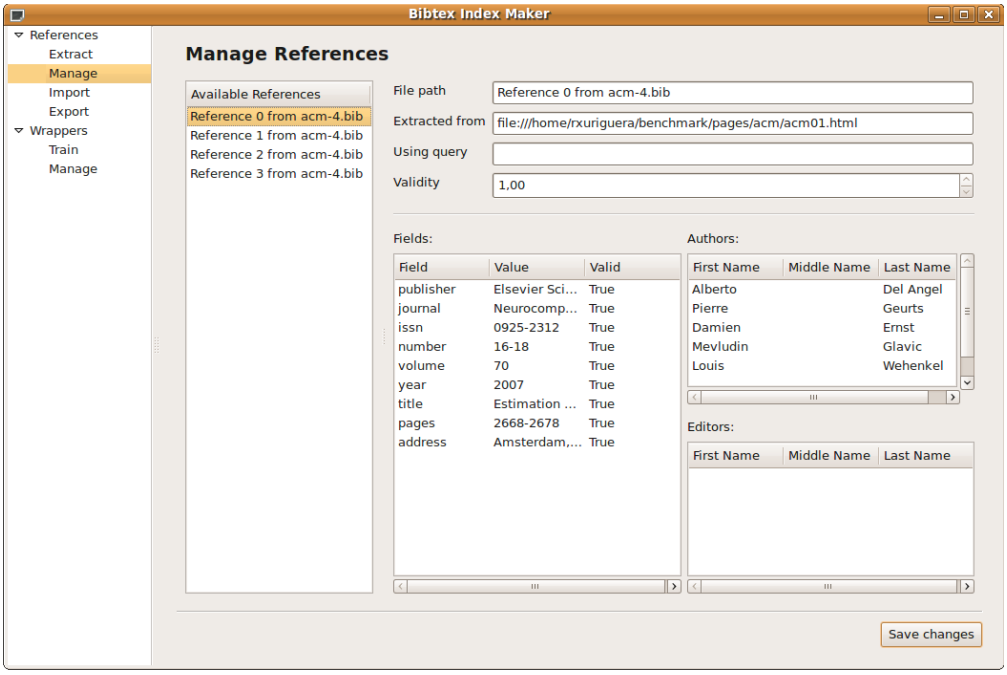


Figura E.2: Maneig de referències

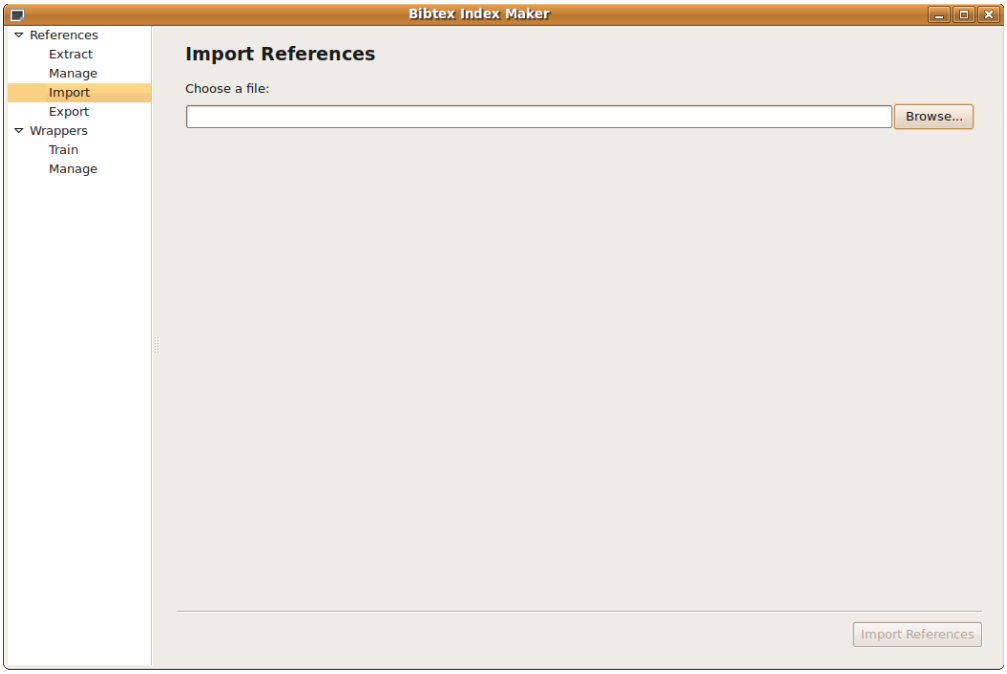


Figura E.3: Importa referències

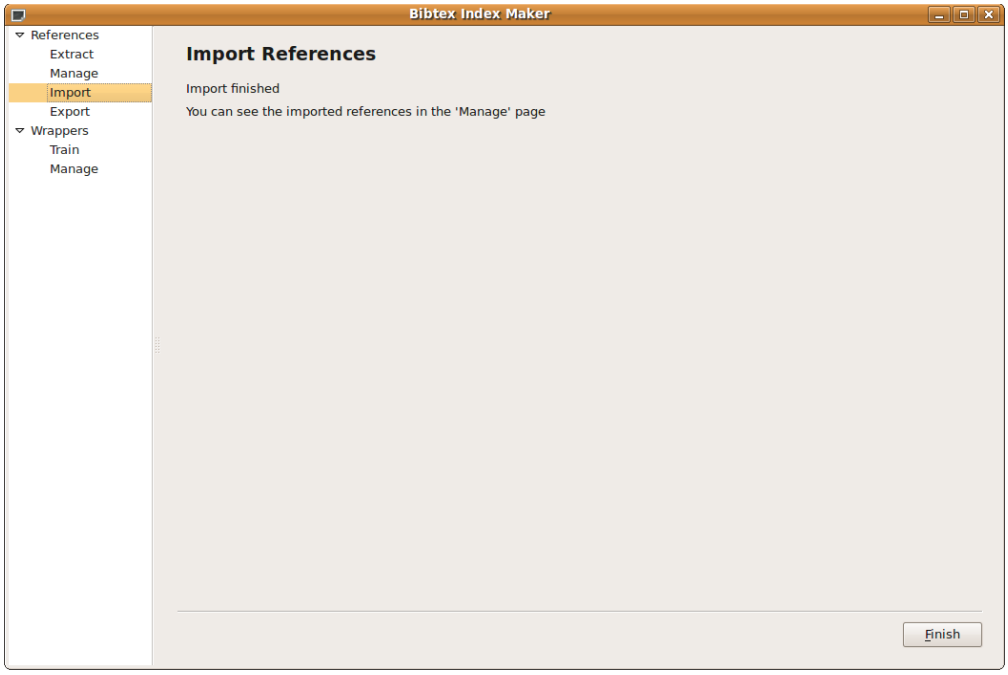


Figura E.4: Un cop s'han importat les referències

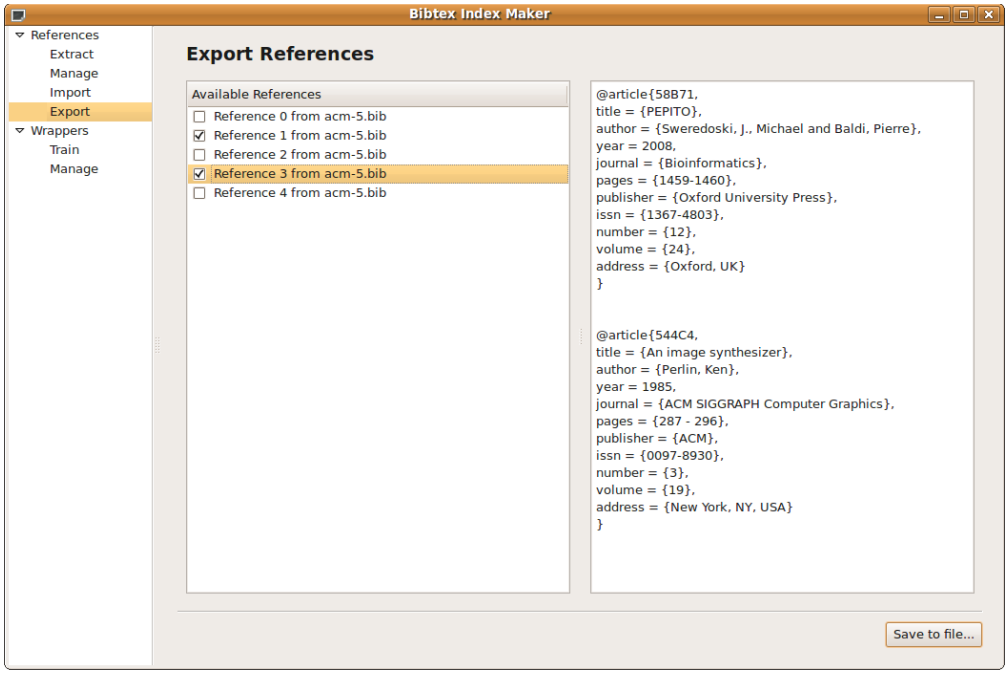


Figura E.5: Exporta referències

E.2 *Wrappers*

L'altra categoria és la que permet gestionar els *wrappers*.

Entrenament

Aquesta opció és la que ens permet generar nous *wrappers* a partir de les referències de les que es disposa. La vista que permet executar aquesta funcionalitat és la de la figura E.6. Les URLs de les biblioteques per les quals ja s'han generat *wrappers* amb anterioritat es mostren a la llista *Available URLs*, facilitant el re-entrenament un cop les dades deixen d'extreure's correctament. En el cas que es tracti d'una biblioteca nova, podem introduir-ne la seva adreça (o un prefix) manualment al camp *URL*.

Un cop hem escollit la biblioteca, només hem de clicar el botó *Train* i deixar que l'aplicació faci la resta. Quan finalitza, es mostra un missatge indicant que podem veure i editar els *wrappers* generats des amb l'opció *Manage*.

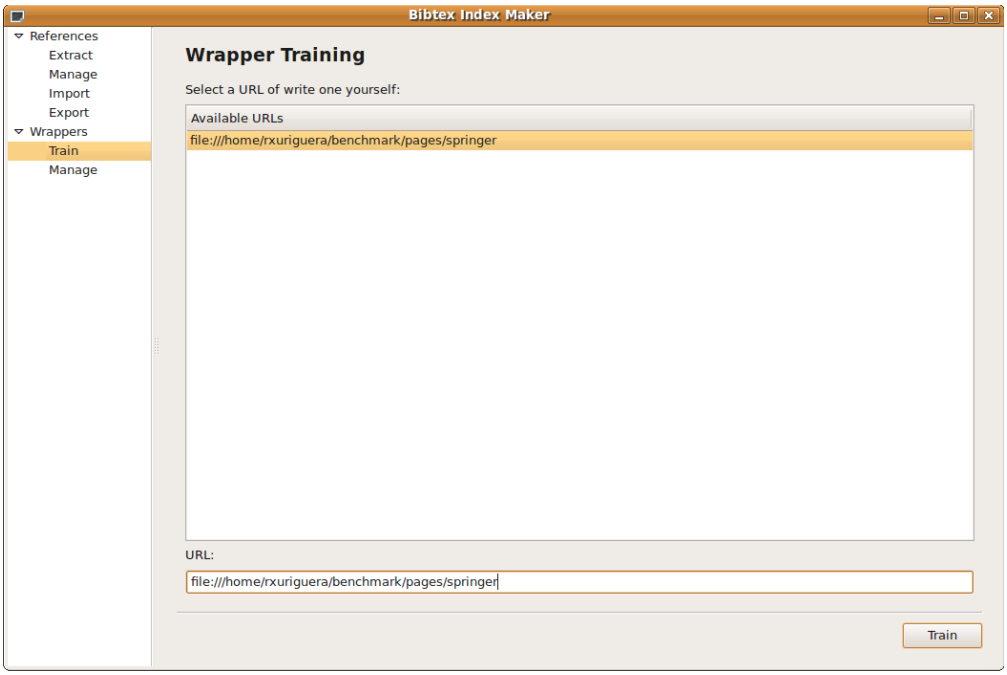


Figura E.6: Entrenament de *wrappers*

Gestió

De la mateixa manera que podem modificar les referències, també tenim un editor pels *wrappers* generats; es mostra a finestra de la figura E.7. En aquest cas tenim tres apartats:

- Llistat de col·leccions de *wrappers* agrupades segons la biblioteca digital a la qual corresponen.

- Llistat de *wrappers* de la col·lecció seleccionada en un moment donat. Per cadascun d'ells es mostra la puntuació rebuda.
- Editor del *wrapper* seleccionat a la llista anterior. Es mostren els vots positius, negatius, la puntuació calculada i el llistat de regles.

Podem crear i eliminar regles afegint nous valors a la línia buida que hi ha al final de la llista *Rules* i deixant línies en blanc, respectivament. Les col·leccions i els *wrappers* es poden crear i esborrar amb el menú contextual que apareix al clicar amb el botó secundari. Aquesta inconsistència a l'hora de realitzar la mateixa operació per elements diferents és un problema que caldrà solucionar en un futur.

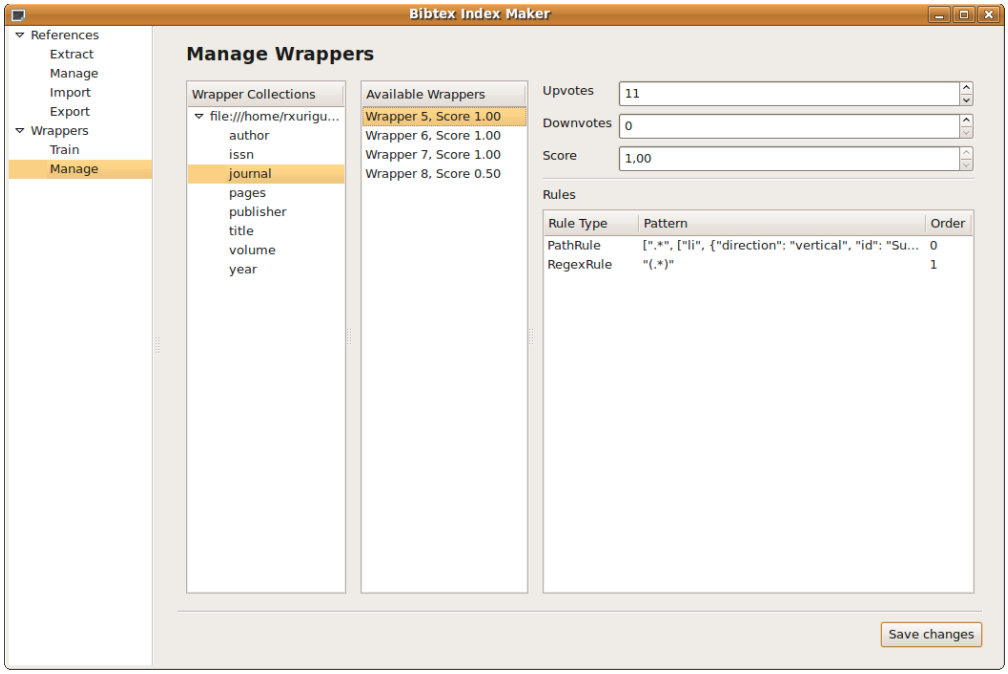


Figura E.7: Gestió de *wrappers*