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

El format PDF ha esdevingut un estàndar 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.
Compta amb tres parts principals:

1.1 Estructura d'aquest document

La memòria està dividida en els següents capítols:

- Capitol 2: Descripció formal del projecte i el seu context així com un repàs sobre el disseny.
- Capitol 3: Parla de les tècniques que s'utilitzen per poder aconseguir pàgines que continguin informació sobre els do

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

- Capitol 4: Tracta sobre com extreure la informació sobre els articles de les pàgines d'Internet que hem obtingut.
- Capitol 5: Està dedicat a les tècniques per generar, de forma automàtica, les regles d'extracció de les referències.
- Capitol 6: Plantejament de les proves per cadascuna de les parts més importants del sistema i anàlisi dels resultats obtinguts.

Pel que fa al contingut de les diferents seccions, ens hem volgut centrar, sobretot, en el que fa a les decisions que hem hagut de prendre al llarg de la realització del projecte i no tant en els aspectes tècnics de com s'ha implementat el sistema. Per cada decisió es presenten algunes de les opcions plantejades en un principi, els problemes que aquestes presenten i es passa a descriure la solució escollida i els motius de l'elecció.

Capítol 2

Definició del Projecte

2.1 Context

S'assumeix que es tenen coneixements bàsics d'HTML i d'expressions regulars.

2.2 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.

2.3 Funcionalitats

La llista completa de característiques del sistema:

- Extracció de la referència bibliogràfica corresponent a un article d'un fitxer PDF
- Possibilitat d'exportar les referències extretes en format BibTeX i desar-les a un fitxer `.bib`

2.4 Disseny del sistema

Hem organitzat el codi del sistema en els mòduls que 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 disponibles a Internet 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*: Per una banda fa un anàlisis sintàctic de les referències extretes per poder-les validar. Per l'altra, transforma a BibTeXles referències extretes.
- Base de dades (db): Tal i com indica el seu nom, duu a terme els accessos la base de dades.
- *Main*: Enllaça tots els mòduls anteriors i proporciona punts d'entrada a la interfície d'usuari. Fa de façana del sistema.
- *Graphical User Interface* (gui): Interfície d'usuari més o menys amigable.

La figura 2.1 mostra com interaccionen entre ells.

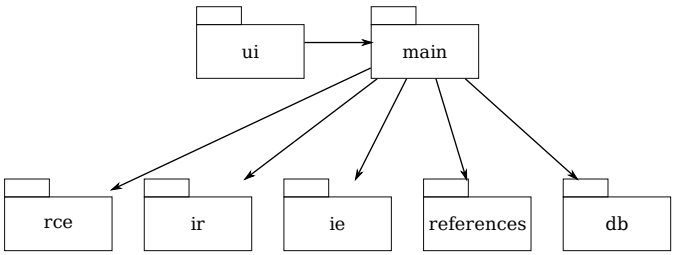


Figura 2.1: Mòduls del sistema

Capítol 3

Cerca de referències

3.1 Extracció dels continguts d'un PDF

El primer pas per aconseguir els objectius proposats és l'extracció del contingut d'un fitxer PDF. Aquest és un dels aspectes que han influït més en l'enfocament que hem donat al sistema, pels motius que es descriuen a continuació.

En un principi, la solució que vam plantejar va ser intentar extreure la referència bibliogràfica d'un document directament del fitxer PDF del qual es disposa. Tot i que a simple vista ja es veu que això pot tenir limitacions (e.g. informació que no es troba dins del text), després de veure com queden els articles al convertir-los a text ens vam allunyar encara més d'aquesta idea. 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, no hi ha cap tipus d'estructura que pugui deixar intuir quina part del text correspon a cada fragment d'informació que ens interessa.

```
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

```
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

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 que es descriuen a continuació. Les principals dificultats que es troben a l'hora d'obtenir el text són:

- Caràcters especialscom ara Unicode o lligadures (e.g. *fi* es representa com un sol caràcter)
- Sub/Superíndexs: la majoria d'eines els extreuen com un número 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
- 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.
- Fitxers protegits dels quals no es pot extreure el contingut
- Documents escanejats:Aquests fitxers només contenen imatge i, no en podem extreure el contingut amb les eines que hem provat.

A banda d'aquestes dificultats tècniques també hi influeix el fet que hi ha un número força reduït de programari lliure que ofereixi aquesta funcionalitat.

3.1.2 Programari

Algunes de les opcions que hem tingut en compte a l'hora d'escollir una llibreria o aplicació d'extracció de text han estat: *PyPDF*, *PDFMiner* o *PDFBox*, tot i que finalment ens hem decantat per *xPDF*.

xPDF és 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 la qualitat dels resultats. En especial, el fet que no separa els paràgrams 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.

3.2 Consultes

El punt més important per poder cercar referències bibliogràfiques a Internet és ser capaços de generar consultes que retornin bons resultats. Una primera idea pot consistir en cercar segons el títol de l'article del qual volem informació. El problema és que bona part dels resultats corresponen a pàgines que fan una referència a aquest article, però que no en donen gaires detalls. Com que a l'extreure el text del PDF la resta de les dades de la capçalera queden desfigurades, és difícil itzar-les per fer les consultes.

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 molts articles el primer paràgraf va precedit de la paraula *Abstract*, també n'hi ha molts altres que van precedits 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 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 un cert nombre que anirem 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 les consultes 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"
- "o 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-ho, hem definit un altre paràmetre que defineix el nombre de consultes a saltar-se des del principi de l'article.

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 principal problema que hem trobat amb aquests és que no han publicat cap API per tal de permetre les consultes automàtiques des d'aplicacions de tercers. Tot i que hi ha solucions a aquest problema, 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 i hem preparat la nostra aplicació per tal d'utilitzar les APIs de *Google*, *Yahoo* i *Bing*. 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.

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 eles 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 ldel tipus de fitxers dels que disposem la qualitat dels resultats obtinguts amb els cercadors poden 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, articles populars 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?id=id-autor.

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 auqestes proves no siguin gaire riguroses, són suficients per poder-nos fer una idea força clara sobre la millora que s'obté utilitzant múltiples fils respecte no fer-ho.

Sobre 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 màxim dependrà 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. En aquest capítol tractarem dels problemes que hem trobat a l'hora d'extreure la in

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.

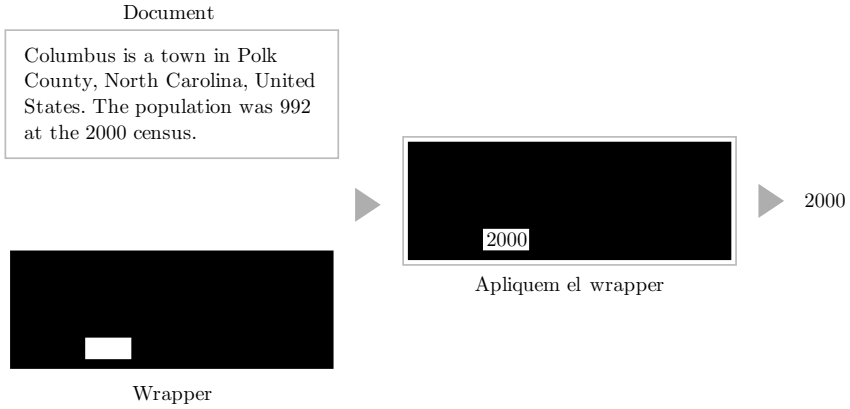


Figura 4.1: Exemple de la funció d'un *wrapper*

A la nostra aplicació tindrem els dos tipus de *wrapper* que es descriuen a continuació.

4.1.1 *Field Wrappers*

S'encarreguen d'extreure únicament un dels camps de la referència cada vegada. Se'n necessita un per cadascun dels camps que volem extreure per cada biblioteca d'articles que vulguem suportar.

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.

En un principi vam començar a definir aquestes regles manualment, mirant de fer-les prou generals com per poder-les reutilitzar per múltiples biblioteques, però de seguida vam adonar-nos que aquesta és una tasca que consumeix molt temps. A més, els resultats es veuen afectats per qualsevol canvi en l'estructura de les pàgines font. Per tant, vam decidir centrar els recursos en trobar la manera de generar aquests *wrappers* de forma automàtica. Com que aquesta és una de les parts més interessants del projecte, li hem dedicat el capítol 5 sencer.

Pel que fa al procés d'extracció, donat el resultat web d'un article mirem si disposem de *wrappers* a la base de dades de l'aplicació i, en cas afirmatiu, obtenim aquells amb la puntuació més elevada. Per cadascun d'aquests *wrappers* provarem d'extreure el camp de la pàgina. Si veiem el resultat obtingut amb algun d'ells es considera vàlid, pararem l'extracció i passarem al següent camp.

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. D'aquesta manera, en aquelles situacions on la informació està estructurada diferent, també podrem obtenir resultats.

4.1.2 *Reference Wrappers*

Aquest tipus de *wrappers* extreuen el text corresponent a una referència sencera. El gran avantatge que tenen és que habitualment permeten extreure molta més 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.

El principal problema és 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'obtenir-les. Això complica força la generació automàtica. A més, si tenim en compte el fet que d'aquests *wrappers* només en necessitem un per cada biblioteca, fa que automatitzar-ho no ens aporti un benefici gaire gran. Per tant, hem decidit implementar-los manualment.

En algunes ocasions, aplicant una mica d'enginyeria inversa 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:

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

Llistat 4.1: Adreça d'un artice de *ScientificCommons*

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. Els formats que s'ofereixen depenen força del camp de coneixement en què s'especialitza la biblioteca.

4.2 Validació de referències

Parsing de referències

Per poder validar les referències obtingudes a partir dels *reference wrappers*, és necessari analitzar-les sintàcticament per tal de poder validar cadascun dels camps. Per aquest motiu, l'aplicació disposa d'un *parser* de referències en format BibTeX.

4.3 Format de referències

Com a detalls de la implementació, només comentar que tindrem un jerarquia de classes anomenades *generadors* (`Generator`) 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
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.

Capítol 5

Generació de *Wrappers*

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 conté 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 document HTML de la pàgina de 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 el codi 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.

El codi HTML es neteja per treure'n els comentaris, salts de línia, i altres etiquetes 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 es necessiten dos exemples 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 de regles segons les dades d'entrada i sortida que reben i retornen. Per poder concatenar regles, és necessari que la sortida d'una regla sigui vàlida per l'entrada de la següent.

Els detalls respecte com es creen els regles varien depenent de cada tipus de regla, però els dos passos bàsics que se segueixen són els següents:

1. Genera regles per un dels exemples.

2. Fusiona les últimes regles obtingudes amb les anteriors.

Donat que volem extreure informació de documents HTML, les regles que necessitem són: una per localitzar l'etiqueta HTML 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*

Aquest tipus de regles són les que permeten localitzar trossos d'informació dins de la pàgina. Els patrons permeten 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: `[(table, 0), (tr, 1), (td, 0)]` i `[(table, 0), (tr, 0), (td, 0)]`. En realitat, ens interessarà combinar les dues rutes en una de sola i que ens permeti 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, les etiquetes i els atributs dels elements de la ruta. És a dir, només modifiquem el número de germà: si dues rutes tenen el mateix número, el deixem tal i com està; si difereixen, el substituïm amb el valor -1. Si dos patrons no es poden fusionar, acabarem creant *wrappers* diferents amb cadascun d'ells.

Mentre apliquem el patró, si veiem un -1 al número de *sibling* fem una cerca de tots els elements d'aquell nivell de l'arbre que compleixen la resta de condicions enlloc d'escollir-ne un directament. Això implica que enlloc d'obtenir un sol element com a resultat, en molts casos en tindrem més d'un. 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. Es pot definir una expressió regular per cada camp dins del fitxer de configuració.

Com és lògic, en alguns casos ens serà més útil que en altres. Per exemple, si estem intentant extreure l'ISSN d'un article, podem indicar-li l'expressió `(\d{4}-\d{3}(\d|X))`. 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 `(\d4)`, podem restringir més el número de coincidències canviant-la per `(Year\:\ \d4)`

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 part 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 línia 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 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 a la mateixa part de la pàgina 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 `(?:.*)` que amb *Python* indica que es pot acceptar qualsevol cadena, però que cal descartar-la a l'hora de retornar els grups coincidents. 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. L'expressió final resultant queda:

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

Aquest mètode de generar les expressions regulars no està exempt de problemes i es podria millorar per convergir a una expressió més general i fent servir menys 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]). El problema el teníem quan la informació del context també variava entre pàgina i pàgina. Per tant, tot i ser una forma més ràpida de crear les regles, no ens donava resultats prou bons. A la llarga es podria mirar d'utilitzar una aproximació intermitja, agafant el context mínim de l'expressió generada amb el mètode actual. D'aquesta manera reduiríem el número d'exemples necessaris per arribar a expressions regulars prou generals.

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

No obstant, hem de comentar que l'aplicació permet 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.

5.2.3 Regles multi valor

Alguns camps com ara els autors o editors tenen múltiples valors. Per poder-los extreure per separat, hem creat variants de les regles anteriors. Assumim que els múltiples valors corresponents al mateix cam copmliran una de les dues condicions següents:

- Es troben en elements HTML diferents, però són germans o cosins. 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

La tècnica per generar les regles per obtenir els elements HTML és molt semblant a les que ja hem explicat pels camps d'un sol valor. L'única diferència és que cal realitzar *merging* dels patrons dels múltiples valors dins de la mateixa pàgina.

Pel que fa a les *regex rules*, les substituïm per:

- *Separator rules*: Són les encarregades de dividir l'entrada en múltiples valors segons una sèrie de separadors. El seu patró consisteix en una llista amb aquests separadors.
- *Multi-value regex rules*: Acompleixen la mateixa funció 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.

- *Person rules*: Només s'apliquen en el cas dels autors i els editors. Reben el nom complet d'una persona i s'encarreguen de separar-lo les diferents part del nom. Es consideren les parts: *first name*, *middle name* i *last name*. Tenir els noms separats permet que a l'hora de generar la referència BibTeX tots els noms estiguin en el mateix format. 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'}
```

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 a l'hora 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 del quue passa quan comparem *wrappers* molt utilitzats amb altres que no ho han estat tant. Per exemple, si tenim un *wrapper* que ha funcionat moltes vegades durant el passat (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, nombrosos llocs web usen el *Wilson score interval* descrit a [Mil09] per ordenar elements.

Si ens parem a pensar, però, aquest comportament és probablement el que realment busquem. Quan un *wrapper* comença a fallar, és molt probable que ho faci degut a 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. També podem agilitzar-ho donant més vots negatius que negatius.

Capítol 6

Anàlisi de resultats

En aquest capítol es mostren les principals proves realitzades per cadascuna de les tres parts del sistema que hem descrit als capítols anteriors

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 d'un llistat de documents PDF
2. Cercar cadascuna de les consultes amb: *Google*, *Bing* i *Yahoo*
3. Per cadascun dels resultats obtinguts, analitzem si és bo o no
4. Comptabilitzem el número de consultes que han fet falta per obtenir el primer *bon* resultat

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

Una altra qüestió sobre la implementació dels tests, 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'haguem de demanar una sola vegada. A més, en molts casos els resultats corresponen al mateix fitxer PDF del qual estem buscant informació i els hem d'ometre.

Les proves 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.

Tot i que en algunes ocasions els resultats bons retornats per *Bing* o *Yahoo* són superiors en nombre, en general, el cercador *Google* ofereix major cobertura amb resultats sobre més articles.

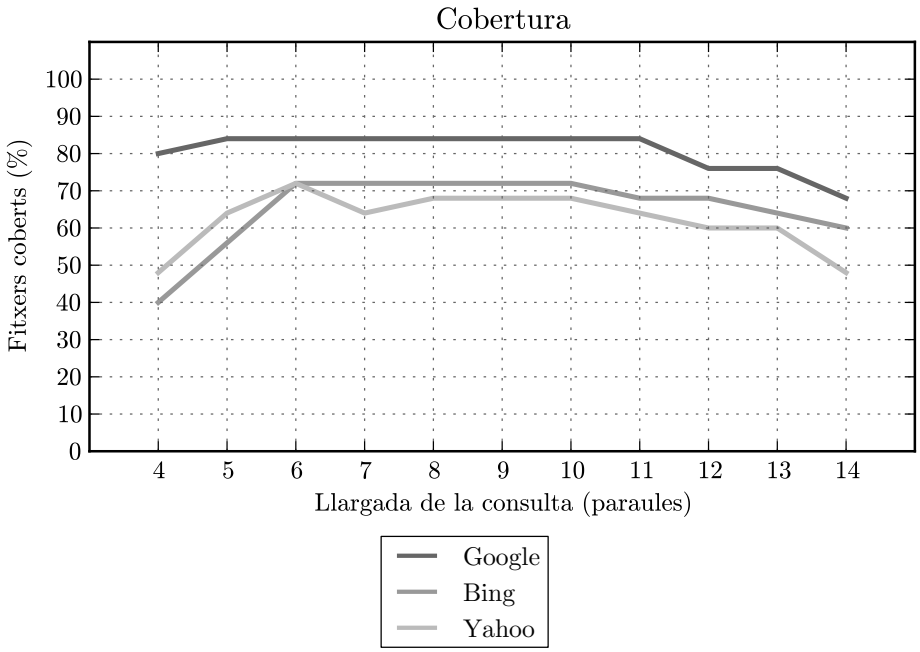


Figura 6.1: Comparació de la qualitat dels resultats obtinguts segons la llargada de les consultes

Finalment, respecte les consultes veiem que com mes llargues són, el número de cerques que hem de fer per començar a obtenir bons resultats disminueix. De totes maneres, tal i com ja s'ha dit a la secció 3.2, per és bo que ens saltem les primeres consultes per evitar cercar amb el títol de l'article.

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. Les pàgines que inclou un grup corresponen totes a la mateixa biblioteca digital. Per cadascun d'aquests conjunts hem importat les referències i hem donat l'ordre al sistema de generar els *wrappers* pels camps de cada biblioteca. Les mostres no són significatives, però ens donen una idea per poder quantificar com de bé funciona l'aplicació dins l'entorn pel qual està pensat.

Els resultats d'aquestes proves corresponen a les puntuacions rebudes durant l'avaluació dels *wrappers* i que 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), les gràfiques no estan generats segons la correctesa dels resultats dels *wrappers* sinó la *confiança* que tenim en que funcionin.

Hem classificat els *wrappers* generats segons quatre grups de confiança, depenent de la puntuació rebuda. La línia fosca correspon al percentatge de les biblioteques digitals per les quals

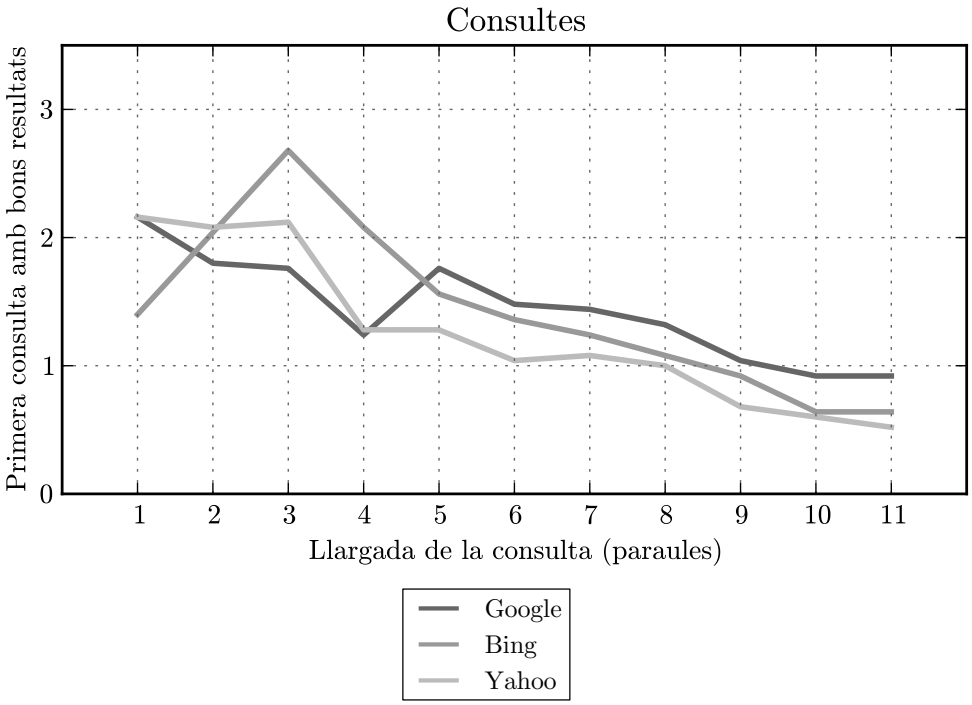


Figura 6.2: Comparació del número de consultes necessàries abans de trobar bons resultats

hem obtingut almenys un *wrapper* de confiança màxima. La línia clara indica el mateix percentatge, però corresponent al següent interval de confiança. Els dos intervals més baixos s'han omès.

Cal remarcar que el fet que per una biblioteca no s'hagi pogut obtenir un *wrapper* que funcioni en tots els casos, no significa que no siguem capaços d'extreure la informació. Tal i com hem vist a la secció 4.1.1, a l'hora d'extreure un camp, es prova més d'un *wrapper*. Per tant, el que ens interessa més veure d'aquestes gràfiques, és la suma dels valors de les dues sèries, que està marcat amb una línia discontinua de gris molt clar.

Com ja s'ha comentat, els camps obligatoris que han de contenir les referències d'articles són: *author*, *title*, *journal* i *year*. A continuació es mostra la confiança que tenim en els *wrappers* per dos d'aquests camps, la informació sobre la resta es poden trobar a l'apèndix B.1.

El camp corresponent a l'any és interessant perquè moltes de les pàgines que hem provat contenen múltiples aparicions del valor que busquem, tot i que no sempre descriuen 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 en aquelles ocasions en què, a més de la informació de l'article, les pàgines inclouen el llistat d'articles als quals la publicació fa referència o bé la llista d'articles que el citen. Quan comencem a tenir exemples on aquests camps tenen valors diferents, al fer *merging* de patrons s'acaben escollint els que són realment vàlids.

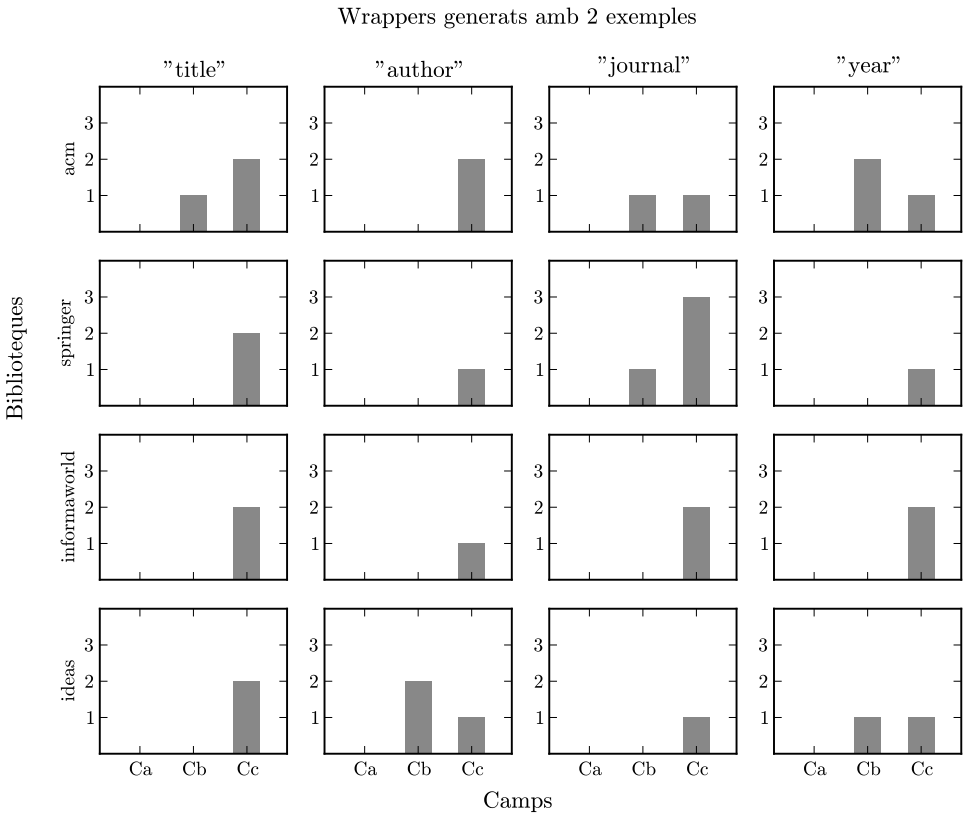


Figura 6.3: Cobertura dels *wrappers* pel camp *journal*

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 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.

6.3 Extracció de referències

text

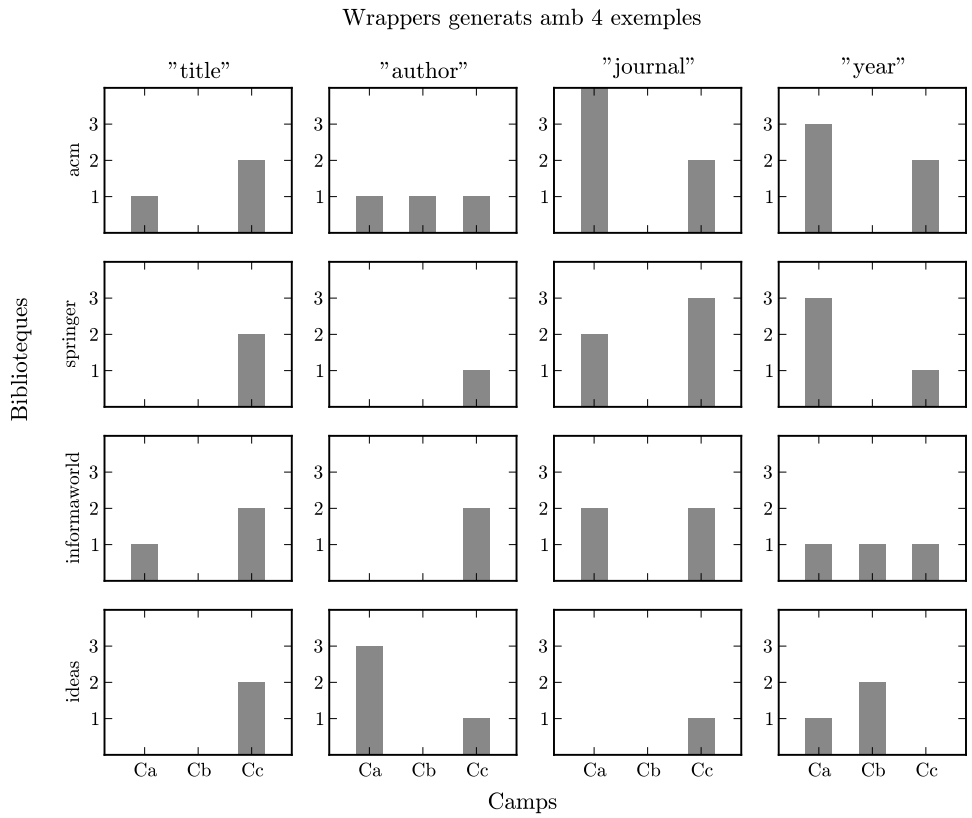


Figura 6.4: Cobertura dels *wrappers* pel camp *journal*

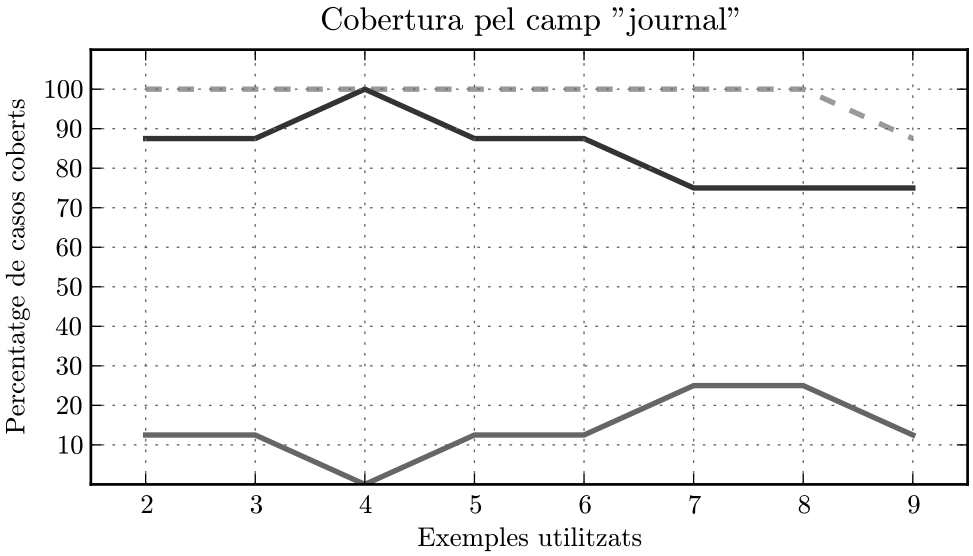


Figura 6.5: Cobertura dels *wrappers* pel camp *journal*

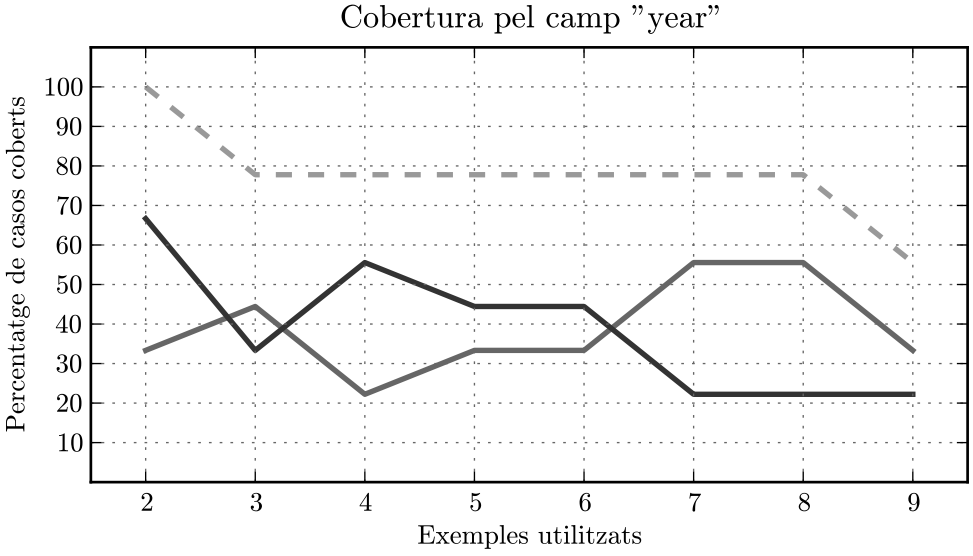


Figura 6.6: Cobertura dels *wrappers* pel camp *year*

Capítol 7

Conclusions i Treball Futur

7.1 Objectius Assolits

7.2 Possibles Millores

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.

[Mil09] Evan Miller. How not to sort by average rating, 2009.

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

Apèndix A

Extracció Contingut PDF

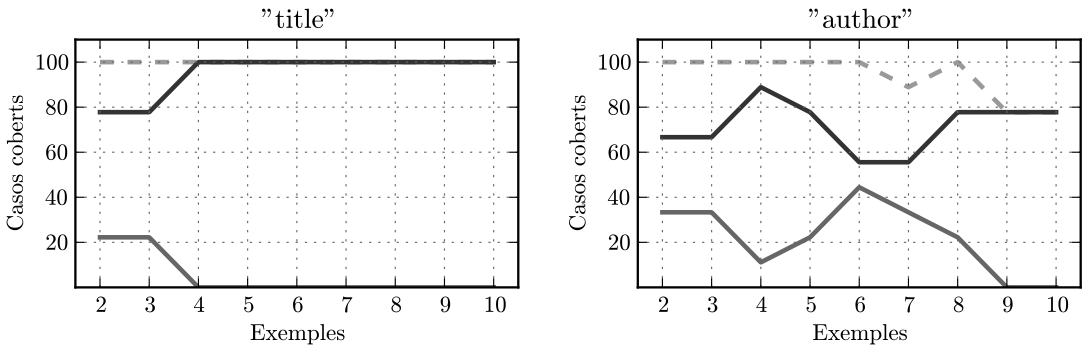
Apèndix B

Resultats dels tests

A continuació es mostren els resultats complets de totes les proves realitzades a la nostra aplicació. L'explicació d'aquests números s'explica al capítol 6.

B.1 Generació de *wrappers*

A continuació es mostren els gràfics de la cobertura dels *wrappers* generats per altres camps que no s'han tractat a la secció 6.2.



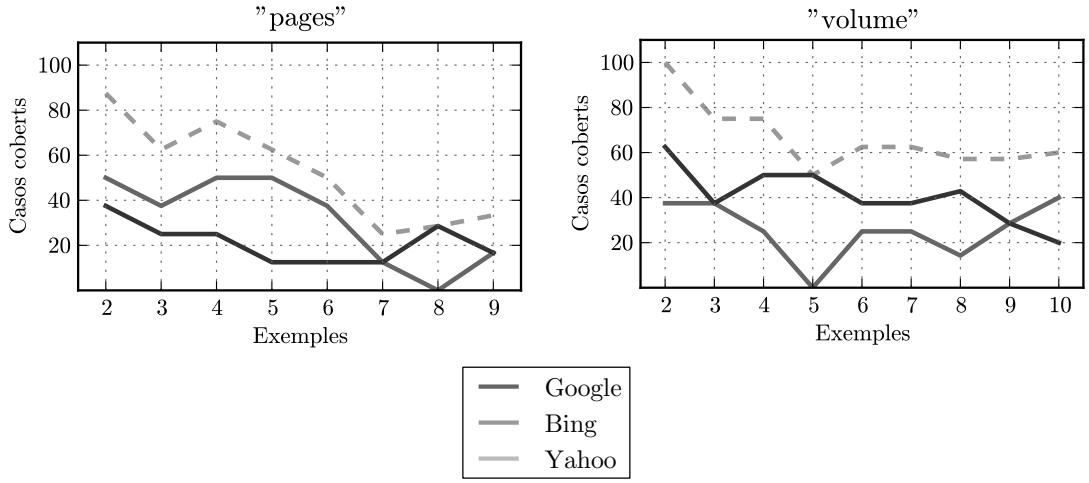


Figura B.1: Cobertura dels *wrappers* generats

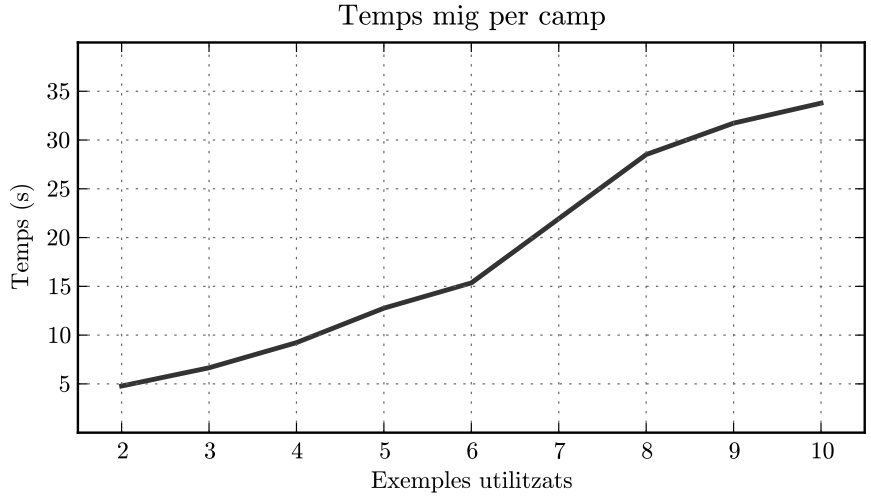


Figura B.2: Temps mitjà per la generació dels *wrappers* d'un únic camp

Apèndix C

Biblioteques utilitzades

Per una banda fen servir els següents mòduls de la biblioteca estàndar. No són els únics que utilitzem, però sí els que ofereixen funcionalitats més interessants:

- RE: Ofereix la possibilitat de treballar amb expressions regulars i amb una sintaxi molt rica. L'hem fet servir àmpliament a tot el sistema.
- SimpleJSON: L'hem utilitzat per dos objectius diferents: llegir fitxers en format JSON 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}'`
- 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 ens facilita les cerques a Internet. L'objectiu del creador era oferir la possibilitat de cercar a *Google* sense 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 els quals podem consultar 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 4.1.2) per tal de poder-les validar.
- PyQt: Es tracta d'una biblioteca en *Python* que embolcalla el framework per la creació d'interfícies gràfiques *Qt*.

Apèndix D

Manual d'usuari