



UNIVERSITAT
ROVIRA I VIRGILI

Departament d'Enginyeria Informàtica i Matemàtiques

Extracció basada en ontologies d'informació de destinacions turístiques a partir de la Wikipedia

TITULACIÓ: Enginyeria Tècnica en Informàtica de Sistemes

AUTOR: Carlos Vicient Monllaó

DIRECTOR: Dr. Antonio Moreno Ribas

DATA: Setembre / 2009.

Agraïments

Vull dedicar aquestes línies als meus pares perquè sense el seu suport incondicional no hagués pogut realitzar els meus estudis.

He d'agrair al Toni Moreno, el meu director, per tots els consells i orientació donats durant els últims mesos.

No voldria oblidar-me de l'Aida Valls, que durant quatre anys ha estat la meva tutora i m'ha oferit la oportunitat de treballar en aquest projecte.

També vull mostrar el meu apleci a tota la resta de membres del Grup de Recerca.

Finalment, gràcies a la meva novia per tota la paciència i ajuda que m'ha donat i per haver estat sempre al meu costat.

INDEX DE CONTINGUTS

0	<u>PRÒLEG</u>	<u>XI</u>
1	<u>INTRODUCCIÓ</u>	<u>1</u>
1.1	MINERIA DE DADES	1
1.2	MINERIA WEB	2
1.3	PROJECTE DAMASK	3
1.3.1	EIXOS D'INVESTIGACIÓ	4
1.3.2	METODOLOGIA I PLA DE TREBALL	5
1.4	AQUEST TREBALL	6
1.4.1	OBJECTIUS I ORGANITZACIÓ	6
1.4.2	ONTOLOGIA TURÍSTICA	6
1.4.3	EINA TRACTAMENT DE MATRIU	7
1.4.4	SISTEMA ANALITZADOR	8
2	<u>ONTOLOGIES</u>	<u>9</u>
2.1	CONCEPTES GENERALS	9
2.2	LLENGUATGES DE REPRESENTACIÓ D'ONTOLOGIES	11
2.2.1	CREACIÓ. COMPONENTS DEL LLENGUATGE OWL	13
2.2.1.1	RDF Schema	13
2.2.1.2	Igualtat i desigualtat	13
2.2.1.3	Característiques de les propietats	14
2.2.1.4	Restriccions de les propietats	14
2.2.1.5	Operadors lògics	15
2.2.2	EDICIÓ. EINES	16
2.3	ONTOLOGIES EXISTENTS EN L'AMBIT TURISTIC	16
2.4	ONTOLOGIA TURÍSTICA	18
2.4.1	PROCÉS DE DESENVOLUPAMENT	18
2.4.2	ONTOLOGIA CREADA (TOURISMOWL.OWL)	22
3	<u>LA WIKIPEDIA</u>	<u>29</u>
3.1	CONCEPTES GENERALS	29
3.2	ESTRUCTURA DE LA WIKIPEDIA	29
3.3	METADADES	32
3.4	INFORMACIÓ ESTRUCTURADA	32
3.4.1	WIKIPEDIA TEMPLATES	35
3.5	EDITAR LA WIKIPEDIA	36
3.6	CAS PARTICULAR SOBRE L'ESTRUCTURA D'UN WIKI DE DESTINACIONS TURÍSTIQUES	37
3.7	RECUPERACIÓ I PARSING (MEDIAWIKI API)	39

4	DISSENY DEL SISTEMA DESENVOLUPAT	40
4.1	EINA TRACTAMENT DE MATRIU	40
4.1.1	UML	40
4.1.1.1	Application	41
4.1.1.2	Interface	42
4.1.1.3	Tree	43
4.1.2	SELECCIÓ I DES-SELECCIÓ DE NODES	45
4.1.3	FUNCIONAMENT	46
4.2	SISTEMA ANALITZADOR	47
4.2.1	VISIÓ GENERAL	47
4.2.2	ANALITZADORS (PACKAGE ANALIZATOR)	47
4.2.2.1	Analitzador de text (<i>AnalizatorText</i>)	51
4.2.2.2	Analitzador de categories (<i>AnalizatorCategories</i>)	57
4.2.2.3	Analitzador d'imatges (<i>AnalizatorImages</i>)	61
4.2.2.4	Analitzador de taules (<i>AnalizatorInfoBox</i>)	65
4.2.3	LECTORS (PACKAGE READER)	67
4.2.4	PROGRAMA PRINCIPAL (PACKAGE MAIN)	69
4.2.5	ÚTILS (PACKAGE UTILS)	71
4.3	LIMITACIONS GENERALS	73
5	RESULTATS I JOC DE PROVES	74
5.1	RESULTATS	74
5.2	INTERPRETACIÓ I CONCLUSIONS SOBRE ELS RESULTATS	78
6	CONCLUSIONS I TREBALL FUTUR	79
6.1	CONCLUSIONS	79
6.2	TREBALL FUTUR	80
7	BIBLIOGRAFIA	81
8	ANNEX	84
8.1	MEDIAWIKI API	84
8.2	TOURISMOWL.OWL	95

LLISTA DE FIGURES

Figura 1 Arquitectura actual de la <i>DBPedia</i> [47].....	VII
Figura 2 Fases del procés d'anàlisi de dades[16].....	2
Figura 3 Classificació de la mineria Web[16].....	3
Figura 4 Eina tractament de matriu, Mode de funcionament A.....	7
Figura 5 Eina tractament de matriu, Mode de funcionament B.....	7
Figura 6 Sistema Analitzador (Mineria de dades+Coneixement semàntic).....	8
Figura 7 Classificació d'Ontologies segons Guarino[15].....	10
Figura 8 Taxonomia e-tourism.....	16
Figura 9 Relacions a l'ontologia d'e-tourism.....	17
Figura 10 Taxonomia de OnTourist.....	17
Figura 11 Relacions a l'ontologia d'OnTourist.....	18
Figura 12 Procés teoric per a la creació d'ontologies.....	18
Figura 13 Procés real (o iteratiu) per a la creació d'ontologies.....	18
Figura 14 Tipus de dades per a les Propietats de <i>City</i>	21
Figura 15 Dominis i rangs de les <i>Object Properties</i> de l'ontologia dissenyada.....	22
Figura 16 Taxonomia de la classe <i>Museum</i>	23
Figura 17 Taxonomia de la classe <i>Festival</i>	23
Figura 18 Taxonomia de la classe <i>Building</i>	24
Figura 19 Taxonomia de la classe <i>Sport</i>	25
Figura 20 Taxonomia de la classe <i>Landmark</i>	26
Figura 21 Taxonomia de la classe <i>Architecture_Styles</i>	27
Figura 22 Relacions <i>domain/range</i> entre les classes de l'ontologia.....	28
Figura 23 Esquema de la base de dades de la Wikipedia.....	30
Figura 24 Esquema Bloc 8, base de dades Wikipedia.....	31
Figura 25 Exemple de metadades associades a una imatge[32].....	32
Figura 26 Exemple links interns de la Wikipedia.....	33
Figura 27 Categories de la Wikipedia per a Catalunya.....	34
Figura 28 Classificació d'informació estructurada de la Wikipedia.....	34
Figura 29 Exemple d'ús de plantilles.....	35
Figura 30 Estructura de l'article d'una ciutat I.....	37
Figura 31 Estructura de l'article d'una ciutat II.....	38
Figura 32 Dependències entre els paquets de l'eina de tractament de matriu.....	41
Figura 33 Classes i mètodes del paquet <i>application</i>	41
Figura 34 Classes i mètodes del paquet <i>interface</i>	42
Figura 35 Associacions i dependències de la classe <i>MyInterface</i>	43
Figura 36 Classes i mètodes del paquet <i>tree</i>	44
Figura 37 Associacions i dependències de la classe <i>OntologyTree</i>	44
Figura 38 Representació gràfica d' <i>OntologyTree</i> i <i>MyList</i>	45
Figura 39 Menu <i>File</i> de l'eina de tractament de matriu.....	46
Figura 40 Menu <i>Tools</i> de l'eina de tractament de matriu.....	46
Figura 41 Menu <i>Help</i> de l'eina de tractament de matriu.....	47
Figura 42 Herència de classes del paquet <i>analizators</i>	48
Figura 43 Mètodes públics, protegits i privats de la classe <i>IAnalizator</i> i <i>Analizator</i>	49
Figura 44 Llegenda diagrames de flux.....	49
Figura 45 Exemple del procés d'extracció d' <i>AnalizatorText</i>	51

Figura 46 Diagrama de flux mètode <i>analyzeNode</i> d' <i>AnalizatorText</i>	52
Figura 47 Diagrama de flux mètode <i>recursiveAnalysis</i> d' <i>AnalizatorText</i>	53
Figura 48 Diagrama de flux mètode <i>analyzeNode</i> d' <i>AnalizatorText</i>	54
Figura 49 Diagrama de flux mètode <i>analyzeWithProperties</i> d' <i>AnalizatorText</i>	55
Figura 50 Diagrama de flux mètode <i>recursiveAnalysisProperties</i> d' <i>AnalizatorText</i>	56
Figura 51 Exemple del procés d'extracció d' <i>AnalizatorCategories</i>	58
Figura 52 Diagrama de flux del mètode <i>extract</i> d' <i>AnalizatorCategories</i>	59
Figura 53 Diagrama de flux del mètode <i>analyzeNode</i> d' <i>AnalizatorCategories</i>	60
Figura 54 Resultat generat per una petició a la Wikipedia que demana els noms de les imatges de l'article de Tarragona	61
Figura 55 Exemple del procés d'extracció d' <i>AnalizatorImages</i>	62
Figura 56 Diagrama de flux del mètode <i>extract</i> d' <i>AnalizatorImages</i>	63
Figura 57 Diagrama de flux del mètode <i>recursiveAnalysis</i> d' <i>AnalizatorImages</i>	64
Figura 58 Exemple d' <i>InfoBox</i> en format raw	65
Figura 59 Diagrama de flux del mètode <i>extract</i> d' <i>AnalizatorInfoBox</i>	66
Figura 60 Diagrama de flux del mètode <i>analyzeInfoBox</i> d' <i>AnalizatorInfoBox</i>	66
Figura 61 Herència de classes del paquet <i>reader</i>	67
Figura 62 Diagrama de flux de <i>WikipediaTextReader</i>	68
Figura 63 Mètodes de la classe <i>ExtractorReader</i>	69
Figura 64 Funcionament programa principal	70
Figura 65 Parts necessàries del programa principal	70
Figura 66 Classes del paquet <i>utils</i>	71
Figura 67 Mètodes de la classe <i>Filters</i>	71
Figura 68 Mètodes de la classe <i>Parser</i>	72
Figura 69 Mètodes de la classe <i>Query</i>	72
Figura 70 Mètodes de la classe <i>ProcessXML</i>	73
Figura 71 Resultats obtinguts durant l'anàlisi	76
Figura 72 <i>Precision</i> , <i>Recall</i> i <i>F-Measure</i> dels resultats obtinguts	77
Figura 73 Comparació de <i>precision</i> , <i>Recall</i> i <i>F-Measure</i> de les quatre ciutats	77
Figura 74 Ontologia turística completa	95

PRÒLEG

Motivacions

Avui dia, vivim en una societat caracteritzada per l'excés d'informació degut al creixement d'Internet, fet que s'ha vist incrementat els darrers anys amb l'aparició de les xarxes socials. Aquest excés pot arribar a suposar un gran problema al moment d'extreure el coneixement de la xarxa, ja sigui pel gran volum de dades existents, com per la dificultat de contrastar-les. La tendència actual aposta per la creació d'una nova filosofia de Web, la *Web Semàntica* o *Web 3.0* [1], que intenta afegir coneixement semàntic a la informació que navega per la xarxa per tal de destriar les dades rellevants de les que no ho són mitjançant mètodes de cerca semàntica d'informació [20] que utilitzen consultes basades en ontologies (RDQL[50], RQL, SPARQL[52]). La *DBpedia* n'és un exemple[47].

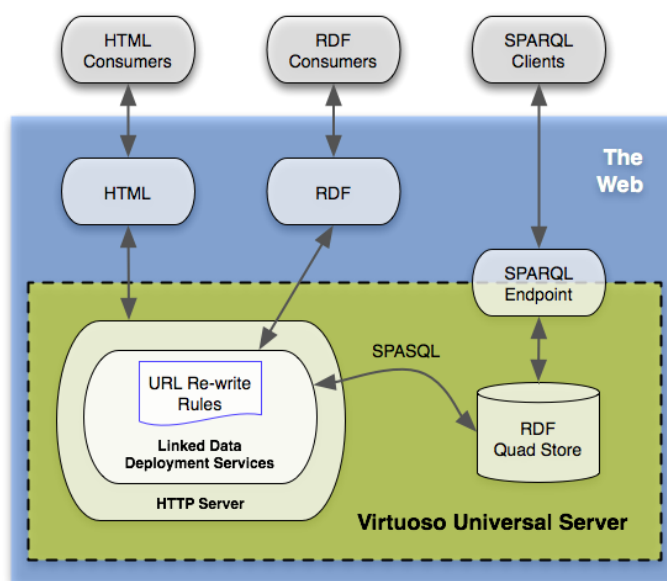


Figura 1 Arquitectura actual de la *DBpedia*[47]

Aquest treball es planteja dintre del marc del projecte de recerca DAMASK- *Data-Mining Algorithms with Semantic Knowledge* (Algoritmes de mineria de dades amb coneixement semàntic), que té com a objectiu principal aportar nous mètodes de mineria de dades guiats pel coneixement semàntic. La mineria de dades neix per l'aparició de noves necessitats caracteritzades per l'augment del volum i varietat de informació que es troba informatitzada en bases de dades digitals.

En moltes situacions, el mètode tradicional de convertir les dades en coneixement consisteix en una anàlisi i interpretació realitzades de forma manual, però això és lent, car i altament subjectiu, ja que és impracticable en dominis on el volum de dades creix exponencialment i desborda la capacitat humana per comprendre-les sense l'ajuda de eines potents. Per tant, l'objectiu de la mineria de dades és resoldre problemes analitzant les dades presents en les bases del coneixement.

La *World Wide Web* és el repositori d'informació més gran i divers dels existents en l'actualitat. La "mineria Web" intenta extreure'n la informació rellevant i útil; aquesta tasca no és un problema senzill: moltes de les dades són no estructurades o semi-estructurades, moltes planes Web contenen dades multimèdia (text, imatges, vídeos, etc.) i moltes d'aquestes dades poden residir en diversos servidors o arxius que afegeixen al problema la dificultat de determinar on buscar i com decidir quina informació és útil. Es

pot veure que una de les limitacions dels mètodes clàssics de mineria de dades és que no solen utilitzar el coneixement del domini; de fet, actualment es reconeix que aquests mètodes produeixen resultats difícils d'interpretar i utilitzar i requereixen que l'usuari faci servir el seu coneixement del domini per poder aplicar-los (p.e. en la presa de decisions)[6].

El projecte on s'emmarca aquest treball proposa la utilització del coneixement semàntic, representat en forma d'ontologies, per definir nous mètodes d'extracció i integració d'informació de recursos Web heterogenis amb diversos graus d'estructuració, classificació i interpretació semàntica dels resultats de la classificació. DAMASK provarà l'aplicabilitat pràctica dels mètodes desenvolupats dintre de l'àrea estratègica del Turisme, que utilitzarà la Wikipedia com a font principal d'informació.

Dins del camp de la Intel·ligència Artificial s'utilitzen les ontologies per representar els coneixements semàntics oferint una projecció estructurada del coneixement d'un domini. Aquestes han passat a tenir un paper fonamental a l'hora d'interpretar continguts textuais i aportar semàntica, al mateix temps que han obert noves vies d'exploració de continguts que fins ara romanien inexplorades.

Organització del document

Aquest document s'estructura en 8 parts ben diferenciades:

- **INTRODUCCIÓ:** En aquesta part s'introdueixen els conceptes de mineria de dades i mineria Web, els problemes que intenten resoldre i les carències dels mètodes actuals. També es presenta el projecte DAMASK i les seves fases i objectius. Finalment, es parla d'aquest treball, que s'engloba dintre del projecte esmentat, i dels objectius que es volen assolir, la feina que es realitzarà i com s'estructurarà.
- **ONTOLOGIES:** Es definirà què és una ontologia, quins són els seus components bàsics (classes, atributs, subclasses, etc.) i amb quines eines es poden crear o manegar. A més a més, també es comentaran algunes de les ontologies existents dintre de l'àmbit turístic. Per acabar, es parlarà del procés de desenvolupament de la nostra ontologia i es mostrarà la seva estructura.
- **LA WIKIPEDIA:** S'explicarà què és i com s'utilitzarà en aquest treball. Es mostrarà l'estructura de les planes Web sobre ciutats, quines són les parts més significatives i importants en les quals ens centrarem i com es realitzarà el procés de recuperació i *parsing* d'aquestes planes.
- **DISSENY DEL SISTEMA DESENVOLUPAT:** S'explicaran tots els aspectes tècnics, mètodes, algorismes, programes i sistemes desenvolupats en aquest treball i la seva forma de funcionar.
- **RESULTATS I JOC DE PROVES:** Es veuran els resultats i estadístiques obtinguts sobre un conjunt de ciutats i es farà un estudi sobre el rendiment i eficiència del sistema.
- **CONCLUSIONS:** Es discutirà quins són els punts forts i febles del treball realitzat, com s'ha aprofitat la semi-estructura dels recursos Web utilitzats, de quina manera es pot generalitzar aquest treball en altres dominis i es farà una valoració final sobre els resultats obtinguts.
- **BIBLIOGRAFIA**

ANNEXOS: Finalment, en els annexos s'afegirà informació complementaria.

1 INTRODUCCIÓ

En aquest capítol es donarà una visió general sobre els temes que es tractaran en aquest document. S'introduiran els conceptes de mineria de dades i mineria Web; també es parlarà sobre el projecte DAMASK i els seus eixos d'investigació, i es finalitzarà el capítol amb una breu presentació sobre aquest treball, els objectius que pretén aconseguir i com s'ha organitzat la feina.

1.1 Minería de dades

La mineria de dades es defineix com el procés d'extreure coneixements útils i comprensibles, prèviament desconeguts, des de grans quantitats de dades emmagatzemades en diferents formats. Així, la tasca fonamental de la mineria de dades és trobar models intel·ligibles a partir de les dades. Perquè aquest procés sigui efectiu hauria de ser automàtic o semiautomàtic (assistit) i l'ús dels patrons descoberts hauria d'ajudar a prendre decisions més segures que reporten, per tant, algun tipus de benefici a la organització.

Per tant, dos són els reptes de la mineria de dades: d'una banda, treballar amb grans volums de dades, procedents majoritàriament de sistemes d'informació, amb els problemes que això comporta (soroll, dades absents, volatilitat de les dades...); de l'altra, utilitzar tècniques adequades per a analitzar-los i extreure'n un coneixement novedós i útil.

De manera senzilla, podríem dir que l'objectiu de la mineria de dades és convertir dades en coneixement.

Sota el nom de “minería de dades” s'engloba tot un conjunt de tècniques encaminades a la extracció de coneixement processable, implícit en les bases de dades. Les bases de la mineria de dades es troben en la Intel·ligència Artificial i en l'anàlisi estadístic. Mitjançant els models extrets utilitzant tècniques de mineria de dades es pot trobar la solució a problemes de predicció, classificació i segmentació; aquestes tècniques són diferents per a cada tipus d'informació.

El procés global d'anàlisi de dades és el procés d'aplicar a una determinada base de dades les operacions de selecció, exploració, mostreig, transformació i modelatge per extreure patrons i, posteriorment, avaluar-los per identificar el conjunt final que representarà el coneixement. Les fases d'aquest procés, com es mostra en la Figura 2, són les següents [16]:

1. Integració i recopilació de dades: Es determinen les fonts d'informació que poden ser útils, es recopilen les dades i es transformen a un format comú per poder analitzar les seves propietats.
2. Selecció, neteja i transformació: s'eliminen o corregeixen les dades errònies i es decideix l'estratègia a seguir amb les dades incompletes. Finalment, es preparen les dades (fusió horitzontal i vertical de files/registres i columnes/atributs) per a poder aplicar la tècnica de mineria que millor s'adapti al problema.
3. Minería de dades: Es decideix la tasca a realitzar (classificació, segmentació, agrupació, predicció, etc.) i es segueix el mètode per construir el model.
4. Avaluació i interpretació: S'avaluen els patrons obtinguts i s'analitzen per experts. Si és necessari, es repeteixen els processos anteriors per millorar els resultats.
5. Difusió: Es fa ús del nou coneixement i s'informa a tots els possibles usuaris.

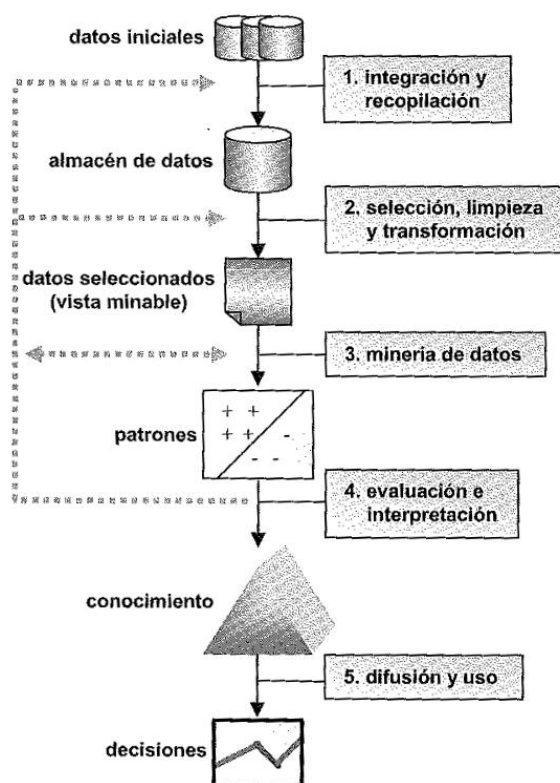


Figura 2 Fases del procés d'anàlisi de dades[16]

1.2 Minería Web

La minería Web es defineix com l'ús de tècniques de minería de dades per descobrir i extreure informació automàticament des de la *World Wide Web*[8].

La minería Web pot descomposar-se en les següents subtasques:

1. **Descobrimient de les fonts:** Localitza i descobreix els documents i serveis de la Web a través de les fonts textuais (hipertextos, correus, etc.) disponibles. El treball de descobrimient es centra principalment en l'ús d'índexs de documents Web.
2. **Selecció i preprocessat de la informació:** Inclou qualsevol procés de selecció i transformació de les dades originals obtingudes en la etapa anterior. La transformació de les dades seleccionades comprèn qualsevol preprocessament, tant orientat a eliminar *stopwords*, etiquetes, etc., com dels destinats a obtenir la representació desitjada (en forma de frases, lògica de primer ordre, etc.).
3. **Generalització: descobrir patrons generals:** Aquesta és la etapa central de la minería Web i és en la que es realitza el procés de minería en sí. La minería Web ha adaptat tècniques de la minería de dades (regles d'associació, agrupament, etc.), de la recuperació d'informació (categorització, classificació de textos), i n'ha desenvolupat algunes de pròpies com, per exemple, l'anàlisi de camins (Web path).
4. **Anàlisi: validació i interpretació dels patrons.** S'encarrega de desenvolupar eines que permetin el consum humà del coneixement minat. Aquestes eines han de incorporar mètodes estadístics i de visualització, així com el coneixement explícit que posseeix l'analista sobre el domini del problema.

Generalment es pot classificar la minería Web en tres àrees d'interès en funció de la part del Web que es mini ([16], Figura 3):

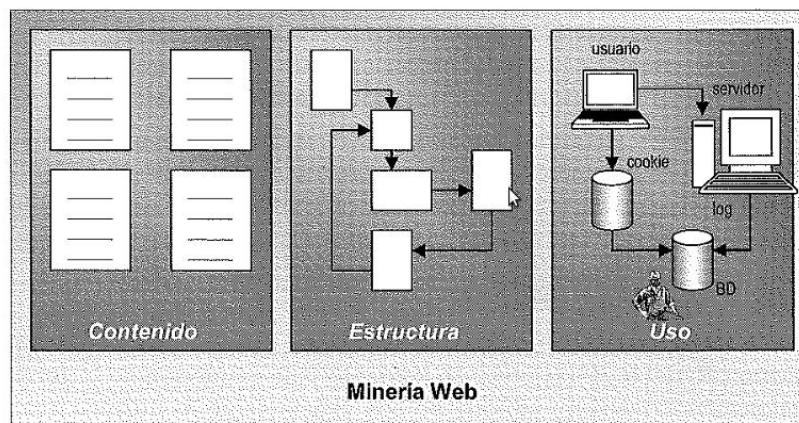


Figura 3 Classificació de la mineria Web[16]

1. Minería del contenido Web: Describe el descubrimiento de información útil de los contenidos textuales y gráficos de los documentos Web y tiene sus orígenes en el procesamiento del lenguaje natural y en la recuperación de la información. Algunas de las datos de la Web están ocultos, ya que se generan dinámicamente o se obtienen en respuesta a peticiones sobre datos que se encuentran en bases de datos privadas. En resumen, los contenidos Web pueden ser de diferentes tipos: textos, imágenes, audio, vídeo, metadatos, hipervínculos, y constan de datos no estructurados como el texto, datos muy poco estructurados como los documentos *html*, datos semi-estructurados como los documentos *xml*, y datos estructurados como las bases de datos.
2. Minería de la estructura Web: Trata de descubrir el modelo subyacente a la estructura de los enlaces de la Web y analiza, fundamentalmente, la topología de los hipervínculos. Esta área puede ser útil para categorizar planes Web y generar información sobre la similitud y relación entre diferentes sitios Web.
3. Minería de uso de la Web: la minería de uso extrae datos derivados de la interacción de los usuarios con la Web para así descubrir patrones en su comportamiento y sus preferencias.

Com s'explicarà posteriorment, en aquest treball s'analitza bàsicament el contingut de les pàgines Web, però també es fan servir els hipervínculs per millorar la qualitat del coneixement obtingut.

1.3 Projecte DAMASK

Com ja s'ha nomenat anteriorment, aquest projecte pretén aportar nous mètodes de minería de dades guiats pel coneixement semàntic d'un domini i, es classifica dintre de la minería del contingut Web.

La motivació principal de DAMASK ve donada per l'evolució actual de les tècniques de minería de dades cap a un model basat en la utilització del coneixement del domini per obtenir millors resultats que facilitin la presa de decisions posterior [30][6].

En aquest context, les hipòtesis de partida que es consideren són:

- H1: El coneixement semàntic del domini pot facilitar l'extracció i integració, de manera eficient, d'informació de recursos Web heterogenis.

- H2: Les tècniques actuals de classificació automàtica de dades poden donar millors resultats si es considera la utilització de coneixement semàntic del domini.
- H3: El turisme és una àrea d'especial interès per a l'aplicació d'aquestes tècniques de mineria de dades basades en informació semàntica, ja que existeix molta informació heterogènia sobre el sector en la Web i un conjunt potencial d'usuaris enorme.

El projecte es centra principalment en l'aplicació de les ontologies en els següents aspectes:

1. El preprocessat de les dades, centrat en la seva adquisició a partir de recursos disponibles de forma lliure i massiva com pàgines Web, la seva integració i la seva transformació en informació processable pels algorismes.
2. Els mètodes de classificació automàtics de les dades, que consideren tot tipus d'informació heterogènia, incloent valors numèrics, categòrics i conceptuals.
3. Els mètodes d'interpretació de les classes obtingudes al pas anterior.

L'objectiu final del projecte és demostrar que la mineria de dades, ajudada de tecnologies semàntiques, ofereix millors resultats i facilita la comprensió d'aquests en comparació amb els algorismes clàssics basats únicament en característiques numèriques o categòriques. Es desenvoluparà un sistema capaç d'agrupar continguts Web semànticament afins que facilitin la seva interpretació.

1.3.1 Eixos d'investigació

El projecte es descomposarà en 3 eixos: processat d'informació Web, *domain-driven data mining* i l'aplicació al turisme.

Processat d'informació Web amb informació semàntica

L'extracció semàntica d'informació es basa en la utilització d'ontologies per a ser capaç d'interpretar el contingut textual d'un recurs independentment del seu format. Encara que hi ha hagut moltes aproximacions conceptuals en el camp de la Web semàntica, en les que es pressuposa disposar de recursos anotats semànticament[21], a mig termini no es pot esperar que l'anotació massiva de recursos Web sigui una realitat. D'aquesta manera, i amb la finalitat d'aprofitar els recursos Web actualment disponibles, l'extracció de característiques a partir de text pla, tal com es pressuposa en el projecte DAMASK, passa per l'anàlisi sintàctic del contingut i la seva associació en respecte als conceptes més adequats d'entre els modelats en una o més ontologies [3].

Respecte al problema de la extracció de característiques rellevants a partir de recursos estructurats o semi-estructurats, existeixen mètodes denominats d'extracció d'informació (*Information Extraction*) que es basen en el fet que el document disposi d'una estructura definida amb l'objectiu de poder interpretar correctament el seu contingut. L'extracció d'informació s'implementa mitjançant la definició d'una sèrie de regles que depenen de l'estructura del document (wrappers, [10]), però té l'inconvenient de la manca d'anàlisi semàntica i la seva total dependència sobre el format del document [8].

Domain-driven data mining

El tractament de dades no numèriques és una tasca típicament associada a la Intel·ligència Artificial. Des de la teoria de conjunts difusos, s'han desenvolupat diverses maneres d'associar coneixement semàntic a termes lingüístics [7].

Els mètodes de mineria de dades també han incorporat aquest tipus de variables als seus algoritmes clàssics, però encara no permeten interpretar el significat dels termes lingüístics. Les ontologies, en canvi, representen les relacions semàntiques entre conceptes, de manera que els termes poden associar-se a conceptes i és possible raonar sobre ells dintre del àmbit d'un domini.

Aquest nou camp d'investigació en *Domain-Driven Data Mining* (DDDM), dins del qual s'emmarca el projecte DAMASK, explora com utilitzar el coneixement semàntic d'un domini per millorar les tècniques d'anàlisi intel·ligent de dades.

Ús d'informació semàntica en l'àmbit turístic

Fins ara, les diferents aproximacions al problema del tractament semàntic d'informació aplicada específicament a l'àmbit turístic pressuposaven recursos textuais pre-anotats semànticament. Aquest aspecte requereix un gran treball manual per part dels productors de continguts i ignora la gran quantitat d'informació desestructurada disponible actualment en la Web.

1.3.2 Metodologia i pla de treball

El projecte DAMASK s'estructura en 3 tasques bàsiques: extracció i integració basada en ontologies d'informació disponible en recursos Web heterogenis (T1), classificació basada en semàntica de les entitats extretes utilitzant algoritmes de clustering (T2) i aplicació dels resultats en l'àmbit turístic (T3). A més a més es prevé una quarta tasca (T4) dedicada a la disseminació i explotació del projecte.

En les dos primeres tasques es realitzen les investigacions bàsiques (estat de l'art, disseny i implementació dels algoritmes) sobre les que es sustenta la tasca T3, que consisteix en la seva aplicació pràctica en l'àmbit turístic.

La tasca T1 consisteix en dissenyar una metodologia per extreure informació de recursos Web i representar-la en forma de matriu atribut-valor. Aquest procés es sustenta en el coneixement representat en una ontologia del domini (especificada en OWL-DL). Es pretén considerar, de forma gradual, recursos amb diferents nivells d'estructuració: des de dades estructurades (p.e. bases de dades on-line) fins a pàgines Web purament textuais, passant per dades semi-estructurades (p.e. Wikipedia). L'integració de les dades obtingudes d'aquests recursos permetrà la construcció d'una matriu que capturi el màxim d'informació rellevant sobre els objectes d'estudi.

La tasca T2 es centra en el disseny d'un mètode de clustering basat en ontologies. Es consideraran com a dades d'entrada la matriu atribut-valor generada en la tasca anterior i l'ontologia utilitzada per encaminar l'extracció de l'informació obtinguda. A partir d'aquesta informació, es dissenyarà una metodologia de *clustering* que utilitzi el coneixement semàntic representat en l'ontologia per guiar el procés de classificació automàtica dels objectes, així com per obtenir una descripció de les classes que permeti la seva interpretació.

En la tasca T3 es vol avaluar l'aplicabilitat dels resultats obtinguts en les tasques anteriors a un cas d'estudi concret: la recomanació personalitzada de destins turístics. Per fer-ho es dissenyarà una aplicació Web que implementi els algoritmes dissenyats i treballi

sobre recursos Web relacionats amb l'àmbit turístic. El procés de *clustering* s'utilitzarà per classificar destins turístics i, tenint en compte les preferències i perfil del usuari, recomanar destins afins.

1.4 Aquest treball

Com ja s'ha comentat anteriorment, aquest projecte s'engloba dintre del projecte DAMASK i, més concretament, es centra en el processat d'informació Web amb informació semàntica. Les tasques a realitzar es basen en les subtasques T1-2 i T1-3 del projecte abans esmentat.

La primera de les tasques consisteix en identificar i analitzar els tipus de dades que s'hauran de manejar i delimitarà amb claredat tots els tipus de valors que es manegaran dintre de la matriu que s'ha de generar. Aquest estudi és necessari per poder definir, a la subtasca T1-3, els mètodes apropiats per adquirir aquesta informació. Es treballarà amb dades numèriques, booleanes i categòriques i amb atributs conceptuals (el valor d'un atribut és el concepte de l'ontologia d'entrada) i textuals (el valor d'un atribut és un terme o llista de termes recuperats de la descripció del objecte a una plana Web textual).

La següent tasca consisteix en dissenyar i implementar noves tècniques d'extracció d'informació, continguda en recursos Web heterogenis que utilitzen coneixement semàntic del domini, representat en una ontologia escrita amb OWL-DL. Aquestes tècniques han de resoldre els inconvenients dels mètodes actuals i han d'estar dissenyades per poder recuperar, a partir de recursos Web amb diferents grau d'estructuració, dades de tots els tipus identificats a la tasca anterior com a rellevants.

1.4.1 Objectius i organització

L'objectiu final d'aquest treball és definir i generar l'estructura de la matriu atribut-valor que permeti la representació d'informació disponible del domini turístic, on cada fila de la matriu correspondrà a un objecte ciutat i cada columna a un atribut. S'utilitzaran com a fonts d'informació un conjunt d'articles Web disponibles sobre destinacions turístiques a través de la Wikipedia (en anglès) intentant aprofitar la semi-estructuració d'aquest recurs.

Per a la realització de les tasques anteriors s'ha dividit la feina en tres parts:

- El disseny d'una ontologia dins el domini turístic.
- El desenvolupament d'una eina que generi una matriu atribut-valor buida donada una ontologia, en la que es pugin seleccionar les classes desitjades.
- La implementació d'un sistema analitzador que tingui com a paràmetres d'entrada els recursos Web analitzables i la ontologia i que, com a resultat, generi una matriu atribut-valor amb els conceptes extrets dels recursos.

1.4.2 Ontologia turística

S'ha fet un petit estudi sobre el turisme per poder determinar quins són els aspectes més importants dintre d'aquest àmbit i conèixer les preferències dels diversos tipus de turistes i els seus perfils. A més a més, s'han consultat altres ontologies turístiques existents [onTourism Ontology[45], e-Tourism Ontology[43], etc.]. S'ha decidit fixar-se en la informació turística relacionada amb els esports, museus, monuments i construccions d'una ciutat (capítol 2).

1.4.3 Eina tractament de matriu

Aquesta eina disposarà de dos modes de funcionament:

- a) Generació de matriu: Permetrà carregar qualsevol ontologia escrita amb OWL-DL i s'utilitzarà per seleccionar totes les classes i atributs desitjats de l'ontologia que hagin d'aparèixer a la matriu. Finalment, es generarà un model de matriu on les columnes seran les classes i atributs seleccionats de l'ontologia. Això significa que es podrà decidir el tipus d'informació que es vol buscar i en quin grau d'especialització. Per exemple, en el cas dels edificis, podem senyalar que només ens interessa quedar-nos amb informació relacionada amb edificis religiosos excloent els altres tipus (comercials, culturals, etc.) o, profunditzant encara més, indicar que, de tots els edificis religiosos, només desitgem conèixer l'existència de mesquites musulmanes d'estil sino-islàmic.

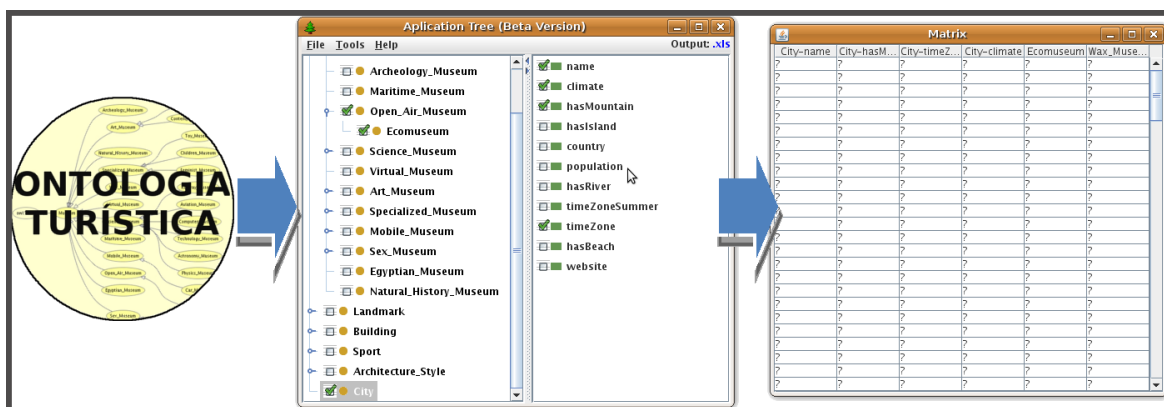


Figura 4 Eina tractament de matriu, Mode de funcionament A.

- b) Tractament de matriu: Permetrà carregar un model de matriu o una matriu generada pel sistema analitzador amb els valors extrets de la Web i transformar-la a diferents formats (.xls, .ser, .properties, .Klass, etc.).

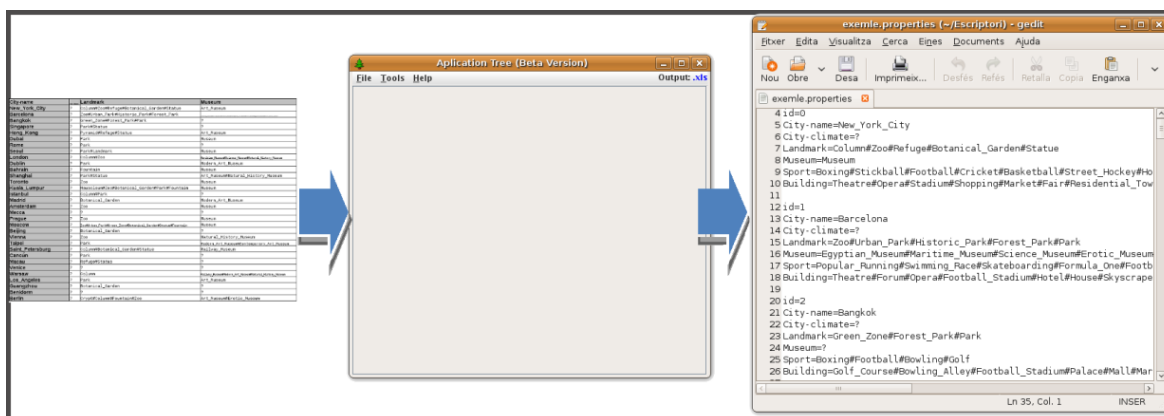


Figura 5 Eina tractament de matriu, Mode de funcionament B

En el capítol 3 es parlarà del disseny d'aquesta eina.

1.4.4 Sistema Analitzador

El sistema analitzador posarà en pràctica les noves tècniques d'extracció d'informació desenvolupades a la tasca T1-3.

Els paràmetres d'entrada al sistema seran, d'una banda, una ontologia escrita en OWL-DL i, de l'altra, els recursos Web que es volen analitzar. D'aquesta manera, l'ontologia reforçarà els mètodes de mineria de dades actuals aportant coneixement semàntic a la cerca.

En aquest treball, només ens centrarem en una petita part dels recursos que es troben a la *World Wide Web*, concretament en recursos semi-estructurats com la Wikipedia.

En la següent pàgina es mostra el funcionament del sistema (Figura 6):

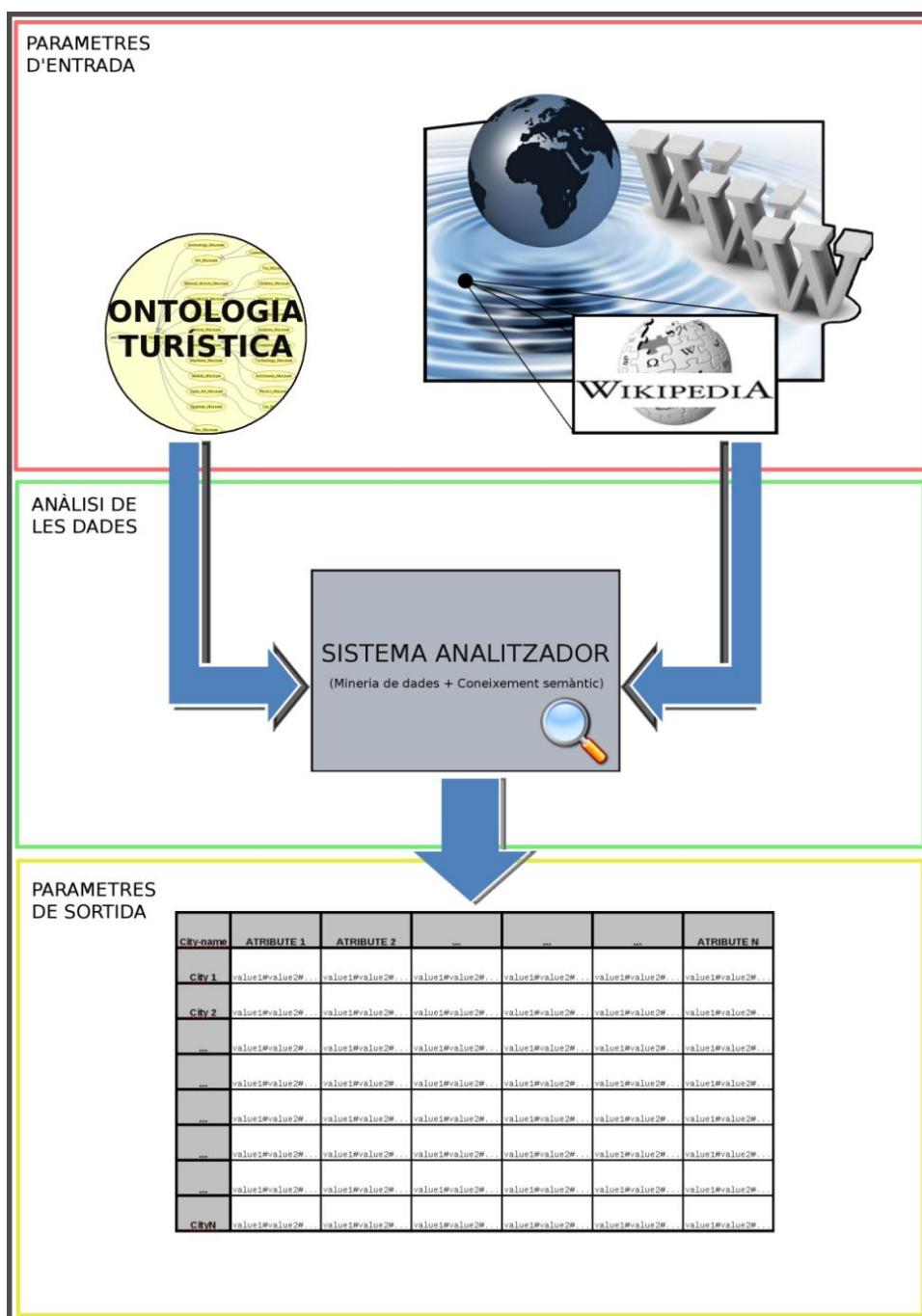


Figura 6 Sistema Analitzador (Mineria de dades+Coneixement semàntic)

2 ONTOLOGIES

En aquest capítol es parlarà sobre què és una ontologia[4] i quins són els llenguatges de representació existents per crear-les. També es nombraran algunes de les ontologies existents dintre de l'àmbit turístic i, finalment, s'explicarà i mostrarà el procés de desenvolupament de l'ontologia creada per a aquest projecte.

2.1 Conceptes generals

Una ontologia es defineix com una especificació formal i explícita d'una conceptualització compartida[25]. *Conceptualització* remet a un model abstracte d'un fenomen en el món en tenir identificats els conceptes rellevants d'aquest fenomen. *Explícita* vol dir que el tipus de conceptes utilitzats, i les limitacions del seu ús, està explícitament definit. *Formal* es refereix al fet que l'ontologia ha de ser llegible per una màquina. Finalment, *compartida* reflecteix la idea que una ontologia captura coneixement consensual, és a dir, que no és privada d'un individu, sinó acceptada per un grup.

Una ontologia també es pot definir segons la seva forma[22]. Una ontologia defineix els termes bàsics i les relacions tot compronent el vocabulari d'un àrea temàtica, així com les regles per combinar els termes i les relacions per definir les extensions al vocabulari. Altres enfocaments han definit les ontologies com a especificacions explícites d'una conceptualització [14] o com una comprensió compartida d'un domini d'interès [26]. Existeixen, doncs, diferents formalismes de representació del coneixement per a la definició d'ontologies. No obstant això, totes elles comparteixen com a mínim els següents components:

- *Classes*: representen conceptes. Les classes solen organitzar-se mitjançant taxonomies a través de les quals es poden aplicar mecanismes d'herència.
- *Relacions*: representen un tipus d'associació entre els conceptes del domini. Les ontologies acostumen a contenir relacions binàries. El primer argument es coneix com el domini de la relació i el segon, com el rang. Les relacions binàries s'utilitzen en algunes ocasions per expressar atributs del concepte. Els atributs es distingeixen de les relacions, perquè el seu rang és un tipus de dades, com ara un String o un número, mentre que el rang d'una relació és un concepte.
- *Instàncies*: s'utilitzen per representar els elements o individus d'una ontologia.

Hi ha diverses classificacions de les ontologies en funció d'un aspecte particular (com ara l'expressivitat[18] o el subjecte i el tipus d'estructura[29]). Una interessant classificació va ser proposada per Guarino[15], el qual classificava els tipus d'ontologies en funció del seu nivell de dependència en una determinada tasca o punt de vista (vegeu Figura 7).

- Ontologies de nivell superior (*Top-level ontologies*): descriuen conceptes molt generals, com “espai”, “temps”, “esdeveniment”, etc., els quals són independents d'un domini particular. Sembla raonable unificar ontologies d'alt nivell per a grans comunitats d'usuaris. Alguns exemples són de Sowa[24], Cyc[19] i SUO[23].
- Ontologies de domini (*Domain ontologies*): descriuen el vocabulari relacionat amb un domini genèric tot especialitzant els conceptes introduïts en la ontologia de nivell superior (*Top-level ontology*). Hi ha diverses ontologies representatives en els àmbits de comerç electrònic (UNSPSC1, NAICS2, SCTG3, i-cat @ ass4, RosettaNet5), Medicina (GALEN6, UMLS7, ON98), Enginyeria (EngMath [13],

PhysSys [2]), Empresarials (Enterprise Ontology [27], Tove [11]) i Gestió del Coneixement (KA [5]).

- Ontologies de tasques (*Task ontologies*): descriuen el vocabulari relacionat amb una tasca o activitat genèrica tot especialitzant les ontologies de nivell superior (*Top-level ontologies*).
- Ontologies d'aplicació (*Application ontologies*): són les més específiques. Els conceptes en les ontologies d'aplicació sovint es corresponen a les funcions desenvolupades per les entitats de domini.

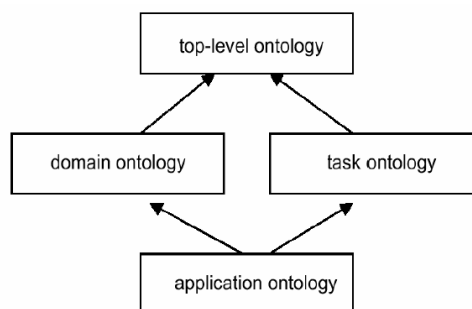


Figura 7 Classificació d'Ontologies segons Guarino[15]

El conjunt d'activitats que s'ocupen del procés de desenvolupament de l'ontologia, el seu cicle de vida, els principis, mètodes i metodologies per a la construcció d'ontologies i el conjunt d'eines i llenguatges que els donen suport s'anomena *Ontology engineering*[12]. Pel que fa a les metodologies, han estat publicades algunes propostes sobre el desenvolupament d'ontologies de forma manual[12].

Tenint en compte la classificació de Guarino, els estudiosos estan sovint involucrats en la tasca de definir les classes bàsiques inalterables i les estructures dels conceptes (objectes, propietats, relacions i axiomes) que són aplicables en tots els àmbits possibles. Aquests principis bàsics estan continguts en les esmentades "Ontologies d'alt nivell" (també anomenades "Fundacionals" o "Ontologies de nivell Superior").

Pel contrari, les Ontologies d'Aplicació tenen un context molt restringit i una reutilització limitada, ja que depenen de l'abast i els requisits d'una aplicació específica. Aquestes ontologies són típicament desenvolupades *ad hoc* pels dissenyadors d'aplicacions. En un punt intermedi, les "Ontologies de Tasca" i les "Ontologies de Domini" són les més complexes de desenvolupar: d'una banda, són prou generals com per ser necessàries per aconseguir el consens entre una àmplia comunitat d'usuaris i, d'altra banda, són prou concretes per presentar una enorme diversitat en dominis de coneixement diferents i dinàmics amb milions de possibles conceptes per modelar. Una iniciativa global com la Web Semàntica depèn en gran mesura d'ontologies de domini. La Web Semàntica [1] tracta d'aconseguir una Web semànticament anotada, el motor de cerca de la qual processaria la informació continguda als recursos de la Web des d'un punt de vista semàntic, augmentant dràsticament la qualitat de la informació presentada a l'usuari. Aquest enfocament requereix un consens global en la definició de les estructures semàntiques adequades (ontologies de domini) per representar qualsevol possible domini del coneixement. En conseqüència, hi ha un ampli acord en el fet que una massa crítica d'ontologies és necessària per a la representació de la semàntica en la Web Semàntica[17].

La construcció d'ontologies de domini depèn de modeladors de domini i enginyers del coneixement que són típicament aclaparats per la potencial grandària, la complexitat i el dinamisme d'un domini específic. En conseqüència, la construcció d'una ontologia de

domini exhaustiva és una barrera que molt pocs projectes poden superar. Resulta que, tot i que les ontologies de domini són reconegudes com a recursos fonamentals per a la Web Semàntica, a la pràctica no estan disponibles, i quan ho estan són poc utilitzades fora de l'àmbit específic de recerca. Per totes aquestes raons, avui dia hi ha una necessitat de mètodes de construcció d'ontologies de domini. En aquest sentit, l'aprenentatge d'Ontologies (Ontology learning) es defineix com el conjunt de tècniques i mètodes utilitzats per a la construcció automàtica d'ontologies des de zero, o bé enriquint o adaptant una ontologia existent d'una manera semi-automàtica, utilitzant coneixement i fonts d'informació distribuïts i heterogenis. Això permet una reducció en el temps i l'esforç necessaris en el procés de desenvolupament de l'ontologia. Com hem dit abans, diversos enfocaments han aparegut durant la última dècada per a l'automatització parcial de l'adquisició de coneixement. Per dur a terme aquest procés, es poden utilitzar alguns dels següents mètodes, tècniques i eines: anàlisi de llenguatge natural, mètodes estadístics, models lingüístics, mineria de text, etc. Aquest procés d'adquisició del coneixement de les dades utilitza el text, diccionaris electrònics, ontologies lingüístiques (com WrdNet[9]) i informació estructurada i semi-estructurada i dades com a fonts de coneixement. Tenint en compte la naturalesa d'aquests corpus d'aprenentatge (abast reduït, lliures de sorolls, fiables, semi-estructurats), els mètodes han estat dissenyats de manera adequada.

En aquest treball ens centrarem en l'ús de les ontologies de domini.

2.2 Llenguatges de representació d'ontologies

Existeixen molts llenguatges per representar el coneixement mitjançant ontologies: RDF (*Resource Description Framework*), OIL (*Ontology Interchange Language*), OWL (*Web Ontology Language*)[51], etc.

Aquest últim és el llenguatge de representació d'ontologies més nou; ha esdevingut un llenguatge de notació estandard arreu del món i ha estat creat per *WebOnt group* amb l'objectiu d'aportar coneixement semàntic a la Web (*Semantic Web*). És una recomanació del W3C¹.

OWL pot ser usat per representar explícitament el significat de termes en vocabularis i les relacions entre aquests termes. OWL té més capacitat per expressar el significat i semàntica que XML, RDF, i RDF-S i, d'aquesta manera, OWL va més enllà d'aquests llenguatges en la seva capacitat per representar contingut interpretable per un ordinador a la Web. OWL és una revisió del Llenguatge de Ontologies Web DAML + OIL i incorpora lliçons apreses a partir del disseny i aplicació de DAML + OIL.

OWL forma part d'un conjunt creixent de recomanacions del W3C relacionades amb la Web semàntica.

- XML: Proporciona una sintaxi superficial per a documents estructurats, però no imposa restriccions semàntiques en el significat d'aquests documents.
- XML Schema: és un llenguatge que s'utilitza per a restringir l'estructura dels documents XML, a més de per ampliar XML amb tipus de dades.
- RDF: és un model de dades per a objectes ("recursos") i relacions entre ells, que proporciona una semàntica simple. Aquest tipus de model de dades pot ser representat en una sintaxi XML.

¹ Consorci World Wide Web (W3C) és un consorci internacional on treballen per desenvolupar estàndards Web. La missió del W3C és guiar la Web cap al seu màxim potencial a través del desenvolupament de protocols i pautes que assegurin el creixement futur de la Web[53].

- RDF Schema: és un vocabulari utilitzat per descriure propietats i classes de recursos RDF permetent la generalització i jerarquitització tant de propietats com de classes.
- OWL: afegeix més vocabulari per descriure propietats i classes, entre d'altres, relacions entre classes, per exemple: “desunió”, “cardinalitat”, “igualtat”, etc. i més tipus de propietats: “simetria”, “classes enumerades”, etc.

OWL proporciona tres llenguatges, cadascun amb un nivell d'expressivitat més gran que l'anterior, dissenyats per ser usats per comunitats específiques de desenvolupadors i usuaris.

- OWL Lite està dissenyat per a aquells usuaris que necessiten principalment una classificació jeràrquica i restriccions simples. Per exemple, encara que admet restriccions de cardinalitat, només permet establir valors cardinals de 0 o 1. Hauria de ser més senzill proporcionar eines de suport a OWL Lite que als seus parents amb major nivell d'expressivitat, i OWL Lite proporciona una ruta ràpida de migració per tesaurus² i altres taxonomies³. OWL Lite té també una menor complexitat formal que OWL DL.
- OWL DL està dissenyat per a aquells usuaris que volen la màxima expressivitat conservant completesa computacional (es garanteix que totes les conclusions siguin computables) i resolubilitat (tots els càlculs es resoldran en un temps finit). OWL DL inclou totes les construccions del llenguatge de l'OWL, però només poden ser usats sota certes restriccions (per exemple, mentre una classe pot ser una subclasse de moltes classes, una classe no pot ser una instància d'una altra). OWL DL és anomenat d'aquesta manera a causa de la seva correspondència amb la lògica de descripció (*Description Logics*, en anglès), un camp d'investigació que estudia la lògica que compon la base formal d'OWL.
- OWL Full està dirigit a usuaris que volen màxima expressivitat i llibertat sintàctica del RDF sense garanties computacionals. Per exemple, en OWL Full una classe pot ser considerada simultàniament com un conjunt de classes individuals i com una classe individual pròpiament dita. És poc probable que qualsevol programari de raonament pugui aconseguir un raonament complet per a cada característica de l'OWL Full.

Cadascun d'aquests subllenguatges són una extensió del seu predecessor més simple, és a dir, les ontologies i conclusions d'OWL Lite són legals en OWL DL i, a la vegada, ho són en OWL Full, però no a la inversa.

Per tant, els components del llenguatge OWL estan formats pels elements que constitueixen els tres subllenguatges. OWL Lite aporta les característiques de l'esquema RDF (*classes, subclasses, individuals, properties, subproperties, domain i range*), que són ampliades en OWL DL i OWL Full.

² En el camp de les Ciències de la Informació, un tesaurus és una llista que conté els "termes" emprats per a representar els conceptes, temes o continguts dels documents, amb la intenció de efectuar una normalització terminològica que permeti millorar el canal d'accés i comunicació entre els usuaris i les unitats d'informació[42].

³ Una taxonomia és una classificació particular, disposada en una estructura jeràrquica. Normalment, està organitzada per relacions pare-fill. Aquest és un tipus de relació d'herència; els subtipus, per definició, tenen les mateixes restriccions que els supertipus més alguna o algunes restriccions addicionals.

2.2.1 Creació. Components del llenguatge OWL

2.2.1.1 RDF Schema

- **Classes** (*Class*): Una classe defineix un conjunt d'individus que pertanyen a un mateix grup perquè comparteixen algunes propietats. Les classes poden organitzar-se en una jerarquia d'especialització usant *subClassOf*. Es pot trobar una classe general anomenada *Thing* que és una classe de tots els individus i és una superclasse de totes les classes de l'OWL.
- **Subclasses** (*rdfs:subClassOf*): Les jerarquies de classe s'han de crear fent una o més indicacions que una classe és subclasse d'una altra. Per exemple, la classe “persona” podria estar definida com subclasse de la classe “mamífer”. D'això podem deduir que si un individu és una “persona”, llavors, també és un “mamífer”.
- **Individus** o instàncies (*individual*): Els individus són instàncies de classes, i les propietats poden ser usades per relacionar un individu amb un altre.
- **Propietats** (*rdf:Property*): Les propietats es poden utilitzar per establir relacions entre individus o d'individus a valors de dades. Es poden utilitzar per relacionar una instància d'una amb un altra instància de la mateixa classe o pot ser usat per relacionar una instància d'una classe amb una instància d'un tipus de dades. Aquests tipus de propietats s'anomenen *Object propertis* i *Data type propertis* consecutivament.
- **Subpropietats** (*rdfs:subPropertyOf*): Les jerarquies de propietats es poden crear fent una o més indicacions que una propietat és al seu torn subpropietat d'una o més propietats.
- **Domini** (*rdfs:domain*): El domini d'una propietat redueix els individus als quals es pot aplicar la propietat. Si una propietat relaciona un individu amb un altre, i la propietat té una classe com a domini, llavors l'individu ha de pertànyer a aquesta classe.
- **Rang** (*rdfs:range*): El rang d'una propietat redueix els individus que una propietat pot tenir com a valor. Si una propietat relaciona a un individu amb un altre, i el valor del rang és una classe, llavors l'altre individu ha de pertànyer a aquesta classe.

2.2.1.2 Igualtat i desigualtat

Permet expressar igualtats o desigualtats entre components ontològics.

- **Classes equivalents** (*equivalentClass*): És possible definir dues classes com a equivalents. Les classes equivalents tenen les mateixes instàncies. El valor d'igualtat pot ser utilitzat per crear classes sinònimes.
- **Propietats equivalents** (*equivalentProperty*): És possible definir dues propietats com equivalents. Les propietats equivalents relacionen un individu amb el conjunt d'altres individus similars.
- **Igual que** (*sameAs*): És possible definir dos individus com iguals. Aquestes construccions es poden utilitzar per crear un nombre de noms diferents que es refereixen al mateix individu.
- **Diferent** (*differentFrom*): És possible definir un individu com a diferent d'altres individus.
- **Tots diferents** (*AllDifferent*): Es pot definir un nombre d'individus com mútuament diferents mitjançant una indicació *AllDifferent*. Aquesta construcció

és especialment útil quan hi ha conjunts d'objectes diferents i quan els dissenyadors estan interessats a reforçar la suposició de noms únics dins d'aquests conjunts d'objectes.

2.2.1.3 Característiques de les propietats

Defineixen la semàntica de les propietats:

- **Propietat inversa** (*inverseOf*): És possible definir una propietat com la inversa d'una altra propietat. Si s'establís la propietat P1 com inversa de la propietat P2, i X es relaciona amb Y mitjançant la propietat P2, llavors Y estaria relacionat amb X mitjançant la propietat P1.
- **Propietat transitiva** (*TransitiveProperty*): És possible definir propietats com transitives. Quan una propietat és transitiva, si el parell (x, y) és una instància de la propietat transitiva P, i el parell (y, z) és una altra instància de la propietat transitiva P, llavors el parell (x, z) també és una instància de P. OWL Lite i OWL DL imposen la condició paral·lela de què les propietats transitives (i les seves súperpropietats) no poden tenir una restricció de cardinalitat màxima (*maxCardinality*) amb valor 1. Sense aquesta condició paral·lela, OWL Lite i OWL DL es convertirien en llenguatges no processables.
- **Propietat simètrica** (*SymmetricProperty*): És possible definir propietats com simètriques. Si una propietat és simètrica i el parell (x, y) és una instància d'aquesta propietat simètrica P, llavors el parell (y, x) és també una instància de la propietat simètrica P.
- **Propietat funcional** (*FunctionalProperty*): És possible definir propietats perquè tinguin un valor únic. Si una propietat és *FunctionalProperty*, aquesta no tindrà més d'un valor per a cada individu (és possible que no tingui un valor per a un individu). Aquesta característica s'anomena propietat única. *FunctionalProperty* és una forma abreujada per indicar que la cardinalitat mínima de la propietat és 0 i la cardinalitat màxima és 1.
- **Propietat funcional inversa** (*InverseFunctionalProperty*): És possible definir propietats perquè siguin funcionals inverses. Si una propietat és funcional inversa, llavors la inversa de la propietat serà funcional. Per tant, la inversa de la propietat té com a màxim un valor per a cada individu. Aquesta característica també s'anomena propietat inequívoca.

2.2.1.4 Restriccions de les propietats

OWL permet establir restriccions sobre la forma en què les propietats són utilitzades per les instàncies d'una classe. Aquests elements (i les restriccions de cardinalitat) es fan servir dins el context d'una restricció d'aquesta manera *owl:Restriction*. L'element *owl:onProperty* indica la propietat restringida. Les restriccions de cardinalitat de l'OWL es refereixen a restriccions locals, ja que es defineixen en les propietats respecte a una classe particular. És a dir, les restriccions forcen la cardinalitat d'aquesta propietat sobre instàncies d'aquesta classe. Les restriccions de cardinalitat de l'OWL Lite són limitades perquè permeten només fer indicacions referents a cardinalitat de valor 0 o 1 (no està permès afegir valors arbitraris per a la cardinalitat, com és el cas en OWL DL i OWL Full).

- **Tots els valors de** (*allValuesFrom*): Aquesta restricció s'estableix sobre una propietat pel que fa a una classe. Això vol dir que aquesta propietat sobre una determinada classe té una restricció de rang local associada a ella. D'aquesta manera, si una instància de la classe està relacionada amb un segon individu

mitjançant aquesta propietat, llavors es pot deduir que el segon individu és una instància d'una classe de la restricció de rang local.

- **Alguns valors de** (*someValuesFrom*): La restricció *someValuesFrom* s'estableix sobre una propietat pel que fa a una classe. Una classe particular pot tenir una restricció sobre una propietat que faci que almenys un valor per a aquesta propietat sigui d'un tipus concret. A diferència de *allValuesFrom*, *someValuesFrom* no restringeix que tots els valors d'una propietat siguin instàncies d'una mateixa classe.
- **Cardinalitat mínima** (*minCardinality*): La cardinalitat s'estableix sobre una propietat pel que fa a una classe particular. Si s'estableix *minCardinality* d'1 sobre una propietat pel que fa a una classe, llavors qualsevol instància d'aquesta classe estarà relacionada almenys amb un individu mitjançant aquesta propietat. Aquesta restricció és una altra manera de dir que és necessari que la propietat tingui un valor per a totes les instàncies de la classe. A OWL Lite les úniques cardinalitats mínimes permeses són 0 o 1. Una cardinalitat mínima de 0 en una propietat indica que la propietat és opcional respecte a una classe.
- **Cardinalitat màxima** (*maxCardinality*): La cardinalitat s'estableix sobre una propietat pel que fa a una classe particular. Si s'estableix *maxCardinality* d'1 sobre una propietat pel que fa a una classe, llavors qualsevol instància d'aquesta classe estarà relacionada com a màxim amb un individu mitjançant aquesta propietat. Una restricció de *maxCardinality* 1 és, de vegades, anomenada una propietat funcional o propietat única.
- **Cardinalitat** (*cardinality*): La cardinalitat es presenta com un avantatge quan és útil establir que una propietat té sobre una classe *minCardinality* 0 i *maxCardinality* 0, o tots dos *minCardinality* 1 i *maxCardinality* 1.

2.2.1.5 Operadors lògics

- **Part de** (*oneOf*): Classes enumerades. Les classes es poden descriure mitjançant l'enumeració dels individus que la componen. Els membres de la classe són exactament el grup dels individus enumerats: ni més, ni menys.
- **Disjunta amb** (*disjointWith*): És possible establir que les classes siguin disjunctes unes de les altres. D'aquesta manera, un raonador pot deduir una inconsistència quan un individu s'indica com a instància de dues classes disjunctes. Això significa que, donades les classes disjunctes C1 i C2, si A és una instància de la classe C1, llavors A no és una instància de la classe C2.
- **Unió amb, intersecció amb i complement de** (*unionOf*, *complementOf*, *intersectionOf*): Combinacions booleans. OWL DL i OWL Full permeten combinacions booleans arbitràries de classes i de restriccions: *unionOf*, *complementOf*, i *intersectionOf*. Utilitzant *unionOf*, podem indicar que una classe conté elements pertanyents a una classe C1 o una classe C2. Utilitzant *complementOf*, podríem indicar que una classe C1 no pot ser mai una classe C2 (és a dir, la classe C1 és una subclasse del complement de C2). Finalment, utilitzant *intersectionOf* es poden establir interseccions entre classes identificades i restriccions. Això és, que un element ha de pertanyer necessàriament a dos classes diferents.

2.2.2 Edició. Eines

Existeixen moltes eines d'edició d'ontologies. Alguns dels editors més importants són OilEd (Universitat de Manchester), OntoWeb (Universitat de Karlsruhe) i Protégé (Universitat d'Stanford).

OilEd no pretén ser un entorn complet per al desenvolupament d'ontologies a gran escala. Està dissenyat per ser el "bloc de notes" d'altres editors d'ontologies. Permet utilitzar el raonador *Fast Classification of Terminologies (FaCT)* per comprovar la consistència de les ontologies, afegir relacions implícites del tipus *subClassOf* i exportar les ontologies a diferents formats com OIL-RDF i DAML-RDF.

OntoEdit és un entorn que dona suport al desenvolupament i manteniment d'ontologies utilitzant eines gràfiques. OntoEdit està construït a la part superior d'un potent model intern d'ontologia. Aquest model intern es pot serialitzar utilitzant XML, el qual recolza la manipulació de fitxers interns. Aquest paradigma dona suport a la representació d'un llenguatge de modelatge neutral en la mesura possible dels conceptes, les relacions i els axiomes.

Protégé és un editor lliure de codi obert i un sistema d'adquisició de coneixement. La plataforma de Protégé soporta dos tipus principals de modelatge d'ontologies a través dels editors *Protégé-Frames* and *Protégé-OWL*. Les ontologies creades en Protégé poden ser exportades a una gran varietat de formats, incloent RDF(S), OWL, and XML Schema. Protégé es basa en Java, és extensible i proporciona un mecanisme de *plugins* que el converteix en una eina flexible i ràpida per a la creació d'ontologies.

Aquest últim, ha estat escollit per a la creació i edició de l'ontologia desenvolupada en aquest treball.

2.3 Ontologies existents en l'àmbit turístic

Abans de dissenyar l'ontologia utilitzada en aquesta feina, es va estudiar la possibilitat d'aprofitar alguna de les ontologies turístiques existents en la Web. Tot i que es van descartar totes, a continuació comentarem dos de les que es van consultar.

La primera d'elles *e-tourism.owl* va ser creada per Katharina Siorpaes per al grup *e-Tourism working group*[44] i pretén descriure els allotjaments i activitats des d'un punt de vista turístic. En la Figura 8, es mostra la taxonomia de l'ontologia i en la Figura 9, les relacions entre les classes. Aquesta ontologia ha estat descartada degut al grau d'especialització tan elevat que posseix, ja que, si s'usés en aquest treball, tan sols ens centrariem en les destinacions turístiques en funció dels allotjaments i activitats de la ciutat en qüestió.

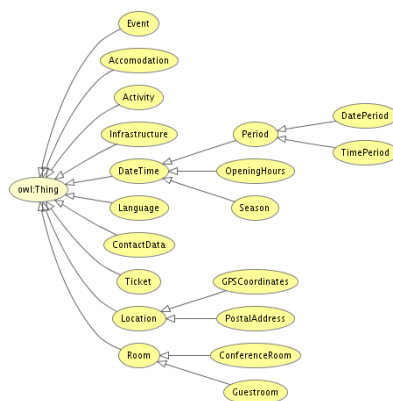


Figura 8 Taxonomia e-tourism

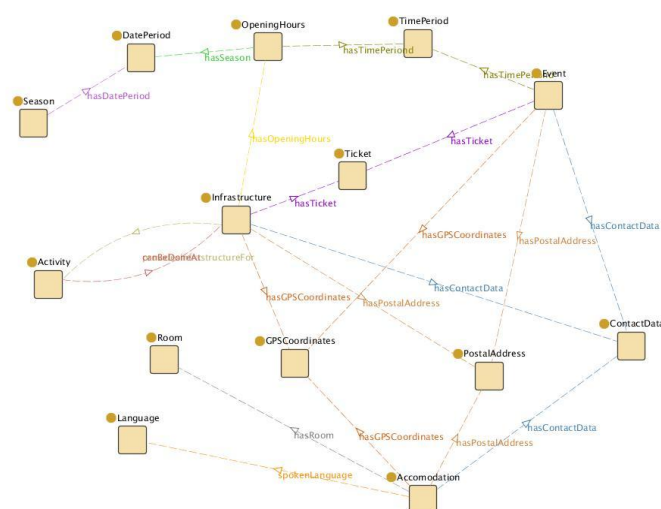


Figura 9 Relacions a l'ontologia d'e-tourism

La segona de les ontologies és OnTourism creada per Michael Luger. En aquesta ontologia es parla de termes com esdeveniments, teatres, cabarets, òperes, festivals, conferències, exposicions, campionats, concerts, esports, esports d'hivern, esports d'estiu, esports extrems, parcs, literatura, fotografia, allotjaments, etc. Es pot observar que aquesta és una opció més atractiva que l'anterior, encara que el seu grau de complexitat és massa elevat. No obstant, s'ha descartat per la gran dificultat de ser interpretada, ja que està escrita en alemany i mostra una taxonomia amb molt pocs nivells de profunditat, fet que presentarà problemes en aplicar les tècniques de *clustering* per poder comparar i classificar les destinacions turístiques. Les següents figures (Figura 10 i Figura 11) mostren la taxonomia i relacions de l'ontologia, igual que en el cas anterior.

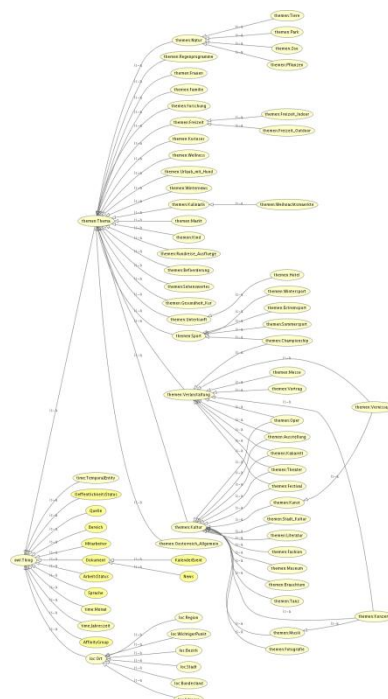


Figura 10 Taxonomia de OnTourist

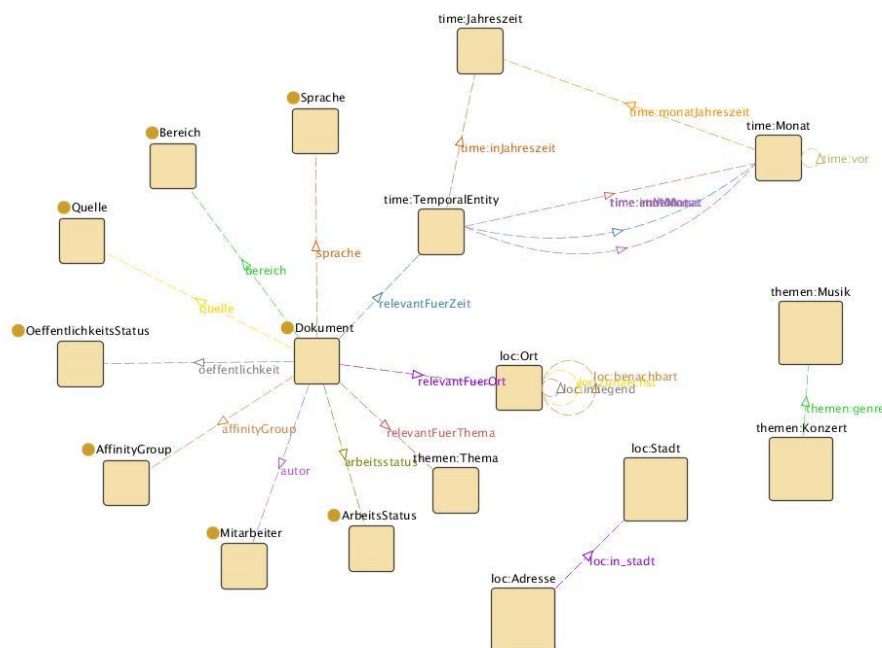


Figura 11 Relacions a l'ontologia d'OnTourist

2.4 Ontologia Turística

Es recorda que l'objectiu d'aquesta ontologia és esdevenir un suport utilitzat per guiar el procés d'extracció semàntic d'informació. En els següents apartats s'explicarà el procés utilitzat per dissenyar i crear l'ontologia i les dificultats que això comporta. Finalment, es mostraran i explicaran totes les parts que la modelen.

2.4.1 Procés de desenvolupament

A nivell teòric, el desenvolupament d'una ontologia està format pels passos que es mostren a la Figura 12. A la realitat, aquest procés és bastant diferent, ja que necessàriament ha de ser iteratiu⁴ (Figura 13).

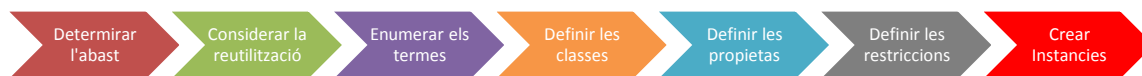


Figura 12 Procés teòric per a la creació d'ontologies



Figura 13 Procés real (o iteratiu) per a la creació d'ontologies

⁴ Relatiu o pertanyent a tot algorisme que obté el resultat desitjat mitjançant l'execució repetida d'una seqüència d'operacions fins a satisfer-se certa condició[33].

Un altre aspecte important a tenir en compte és que els conceptes d'una ontologia han de ser fidels als objectes físics o lògics i les relacions que s'hi representen dintre d'un domini d'interès.

A continuació, es seguiran els passos representats en les figures anteriors per a la creació de la nostra ontologia.

Determinar l'abast

Per determinar l'abast s'ha de contestar quin domini cobrirà l'ontologia, per a què s'utilitzarà o a quin tipus de preguntes es vol trobar una resposta (*competency questions*) mitjançant el seu ús.

És evident que el domini de l'ontologia ha d'estar acotat dintre de l'àmbit turístic i, com ja s'ha explicat, s'utilitzarà per guiar les cerques semàntiques realitzades durant el procés d'extracció d'informació. A més a més, dintre d'aquest domini, es volen contestar preguntes com quants habitants hi ha en una ciutat, quin clima hi fa, quins tipus de turisme s'hi pot fer, quins llocs es poden visitar, quines són les principals atraccions turístiques o quins són els llocs més emblemàtics de la ciutat.

El gran problema dintre de l'entorn turístic es determinar quin tipus d'informació és del nostre interès. Això es degut al fet que no existeix ni una definició ni una classificació estàndard per al turisme, ja que aquest canvia constantment segons els aspectes socials o econòmics i les preferències dels turistes en moments determinats. Actualment, es parla de l'Enoturisme i "Les rutes del vi", per donar un exemple d'activitats inexistents fins fa relativament poc temps i que ara s'han convertit en una tendència.

Ara bé, es podria intentar fer una classificació molt general sobre els tipus de turisme existents basant-nos en la definició que dona la Organització Mundial de Turisme (OMT), que diu que el turisme es defineix com "el conjunt d'activitats que realitzen les persones durant els seus viatges i estances en llocs diferents al seu entorn habitual, per un període inferior a un any, amb fins d'oci, negocis o altres motius".

Considerar la reutilització

Sempre és interessant reutilitzar ontologies ja creades, donat que això ens aporta alguns avantatges: s'estalvia feina, s'evita repetir treballs existents, es poden aprofitar eines que han estat fetes per manipular les ontologies, etc. No obstant, després d'analitzar algunes de les ontologies turístiques (apartat 2.3), s'ha considerat necessari crear una de pròpia per cobrir les necessitats d'aquest projecte.

Enumerar els termes

L'objectiu d'aquest pas és aconseguir una llista amb totes les paraules clau necessàries per contestar a les *competency questions* dins el domini definit.

Per tant, en base als anàlisis dels apartats anteriors i al domini definit, es parlarà de museus i tipus de museus, d'edificis i tipus d'edificis, d'esports i tipus d'esports, d'activitats o festivals, de ciutats i fins i tot d'estils arquitectònics. A la següent llista es mostren les paraules clau que es tindran en compte:

Altitude, Metropolitan Area, Coordinates, Country, Founded, Name, Population Density, Total Population, Time Zone, Total Area, Sport, Aerial Sport, Bridge Jumping, Parachuting, Paragliding, Competition Sports, Swimming Race, Popular Running, Aquatic Sport, Waterskiing, Kayaking, Swimming, Diving, Windsurfing, Water Polo, Rafting, Surfing, Sailing, Terrestrial Sport, Martial Art, Karate, Judo, Boxing, Silat, Dance Sport, Ballet, Nature Sport, Equestrian, Hunting, Mountain Biking, Skiing, Trekking, Climbing, Cycling, Snowboarding, Street Sport, Stickball, Skateboarding,

Motor Sport, Motorbike Sport, Moto Gp, Car Sport, Rally, Formula One, Field Sport, Paddle, Badminton, Cricket, Squash, Handball, Volleyball, Basketball, Baseball, Golf, Tennis, Rugby, Bowling, Hockey, Street Hockey, Roller Hockey, Ice Hockey, Football, Table Tennis, Building, Cultural Building, Library, Opera, Amphitheatre, Natural Amphitheatre, Contemporary Amphitheatre, Roman Amphitheatre, Forum, School, Private School, Business School, Public School, Art School, Music School, Theatre, Ancient Greek Theatre, University, Technological University, Private University, Public University, Religious Building, Miscellaneous Building, Parish, Monastery, Basilica, Sanctuary, Temple, Islamic Building, Mosque, Jewish Building, Synagogue, Christian Building, Chapel, Convent, Abbey, Church, Cathedral, Militar Building, Headquarter, Tower, Prison, Royal Residence, Fort, Palace, Sport Building, Aquatic Centre, Squash Centre, Sport Complex, Velodrome, Golf Course, Pool, Formula One Circuit, Stadium, Rugby Stadium, Hockey Stadium, Cricket Stadium, Baseball Stadium, Football Stadium, Bowling Alley, Industrial Building, Textil Mill, Cotton Mill, Skyscraper, Comercial Building, Shop, Shophouse, Clothes Shop, Fashion Shop, Outlet, Souvenir Shop, Antiquarian Shop, Casino, Casino Resort, Store, Convenience Store, Boockstore, Comercial District, Shopping, Shopping Area, Shopping Centre, Mall, Shopping Street, Comercial Complex, Kiosk, Tianguis, Fair, Fashion Fair, Trade Fair, Artisan Fair, Market, Hypermarket, Supermarket, Food Market, Flea Market, Residential Building, Garden Apartments, Townhouses, Camping, Residential District, Hotel, Luxury Hotel, Residential Tower, House, Landmark, Conmemorate Landmark, Monoliths, Obelisk, Contemporary Obelisk, Obelisk of 21th Century, Obelisk of 17th Century, Obelisk of 18th Century, Obelisk of 19th Century, Obelisk of 20th Century, Ancient Obelisk, Pre-Columbian Obelisk, Keralian Obelisk, Assyrian Obelisk, Ethiopian Obelisk, Egyptian Obelisk, Triumphal Arch, Park, Urban Park, Nature Reserve, Forest Park, Refuge, Garden Park, Botanical Garden, Suburban Park, Historic Park, Zoo, Green Zone, Thematic Park, Tomb, Mausoleum, Pyramid, Sarcophagi, Sepulchre, Crypt, Megalithic, Geographical Landmark, Terrace, Rice Terrace, Square, Mountain, Hill, Cave, Gorge, Memorial Landmark, Statue, Fountain, Grave Stone, Column, Ionic Column, Corinthian Column, Composite Column, Tuscan Column, Doric Column, Water Landmark, River, Canal, Bridge, Stone Bridge, Pedestrian Bridge, Beach, Polder, Lake, Architecture Style, Islamic Style, Mamluk, Sino-Islamic, Indo-Islamic, Persian, Turkistan, Moorish, Sub-Saharan, Ottoman, Fatimid, Victorian Style, Renaissance Revival, Italianeste, Neo Classical, Jacobethan, Victorian, Romanesque Revival, Queen Ane, Painted Ladies, Neo Grec, Second Empire, Gothic Revival, Style, Art Deco, Baroque, Neo Gothic, Mannerism, Modern, Medieval, Renaissance, Byzantine, Romanesque, Festival, Art Festival, Carnival, Christian Festival, Music Festival, Song Festival, Museum, Science Museum, Aviation Museum, Astronomy Museum, Physics Museum, Industrial Museum, Railway Museum, Technology Museum, Computer Museum, Archeology Museum, Art Museum, Contemporary Art Museum, Modern Art Museum, Folk Art Museum, Art Gallery, Sex Museum, Erotic Museum, Maritime Museum, Egyptian Museum, Virtual Museum, Mobile Museum, Car Museum, Specialized Museum, Music Museum, Feminist Museum, Woman Museum, Children Museum, Biographical Museum, Fishing Museum, Military Museum, Toy Museum, Natural History Museum, Wax Museum, Open Air Museum, Ecomuseum, City.

Definir les classes

L'objectiu és descobrir, de la llista anterior de termes, els elements que representen conceptes del domini. Es pot determinar que cada element comprés entre l'element *Sport* i l'element *City* (ambdós inclòsos) de la llista anterior representarà una classe de l'ontologia. Per convenció, el nom de cada classe començarà per majúscula i els conceptes formats per més d'un mot es separaran mitjançant el símbol '_' (per exemple: Art_Museum).

En aquest treball, les classes anteriors constituïran una taxonomia jeràrquica del tipus *Top-down*, on es definiran els conceptes més generals a nivells superiors i s'especialitzaran a cada subnivell.

Definir les propietats

En aquest pas, es descriuen els atributs de les instàncies d'una classe i les relacions amb altres instàncies.

Pel que fa a la classe *city*, es definiran les propietats *altitude*, *metropolitan area*, *coordinates*, *country*, *founded*, *name*, *population density*, *total population*, *time zone*, *total*

area. En aquest cas, les propietats s'escriuran amb la primera lletra minúscula i, les que estiguin formades per més d'un mot, es separaran de la mateixa manera que es fa en les classes. Totes aquestes propietats són del tipus *Data type properties*.

En una versió inicial de l'ontologia es van crear moltes propietats i relacions entre les classes. Més endavant, es va comprovar que, segons les necessitats del problema que es presenta en aquest projecte, no eren necessàries i es van eliminar.

No obstant això, existeix una classe especial, la classe *Architectonical_Style*, que permet relacionar edificis amb els seus estils arquitectònics mitjançant propietats del tipus *Object properties*. L'existència d'aquesta classe és força important, ja que, sense ella, per a cada tipus d'edifici haurien d'existir subclasses que representessin l'edifici més l'estil arquitectònic (fet gairebé impracticable per l'existència de tants estils diferents). Per exemple, la classe *Cathedral* hauria de ser la superclasse de *Gothic_Cathedral*, *Gothic_Revival_Cathedral*, *Romanesque_Cathedral*, *Renaissance_Cathedral*, etc. El mateix passaria amb les classes *Church*, *Chapel*, *Mosque*, i així amb totes les subclasses de *Building*. En l'apartat 2.4.2, es mostraran totes aquestes relacions.

Definir les restriccions

Finalment, per a totes les propietats s'han de descriure els seus límits i determinar quins tipus de valors poden prendre i definir els dominis i rangs de cada propietat. A les següents taules es mostren les restriccions sobre les propietats de la classe ciutat (Figura 14) i els dominis i rangs de les propietats dels edificis (Figura 15).

Propietat	Rang	Propietat	Rang
altitude	Double	name	String
area_metro	Double	population_density	Double
ordinates	String	population_total	Enter
country	String	time_zone	String
founded	String	total_area	Double

Figura 14 Tipus de dades per a les Propietats de *City*

Propietat	Domini	Rang
christianStyle	Christian_Building	Style
		Victorian_Style
islamicStyle	Islamic_Building	Islamic_Style
jewishStyle	Jewish_Building	Style
		Victorian_Style
miscellaneousStyle	Miscellaneous_Building	Style
		Victorian_Style
residentialStyle	Residential_Building	Style
		Victorian_Style
skyscraperStyle	Skyscraper	Style
		Victorian_Style

Figura 15 Dominis i rangs de les *Object Properties* de l'ontologia dissenyada

2.4.2 Ontologia creada (*TourismOWL.owl*)

Finalment, amb l'ajuda del llenguatge OWL-DL i de l'editor d'ontologies Protégé s'ha creat l'ontologia dissenyada a l'apartat anterior (apartat 2.4.1)

Com ja s'ha anat veient, l'ontologia està formada per sis classes principals, de les quals penjen totes les subclasses, i d'una classe secundària per representar els estils arquitectònics. A continuació es mostra l'esquema taxonòmic de l'ontologia:

Classe Museum



Figura 16 Taxonomia de la classe *Museum*

Classe Festival

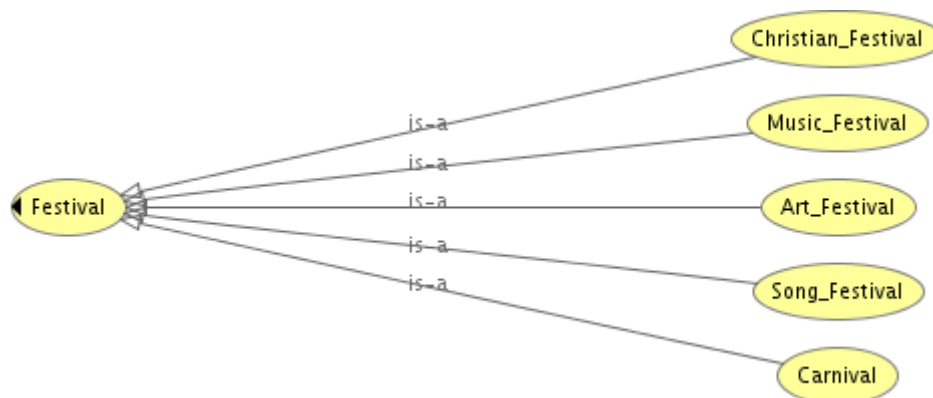


Figura 17 Taxonomia de la classe *Festival*

Classe Building

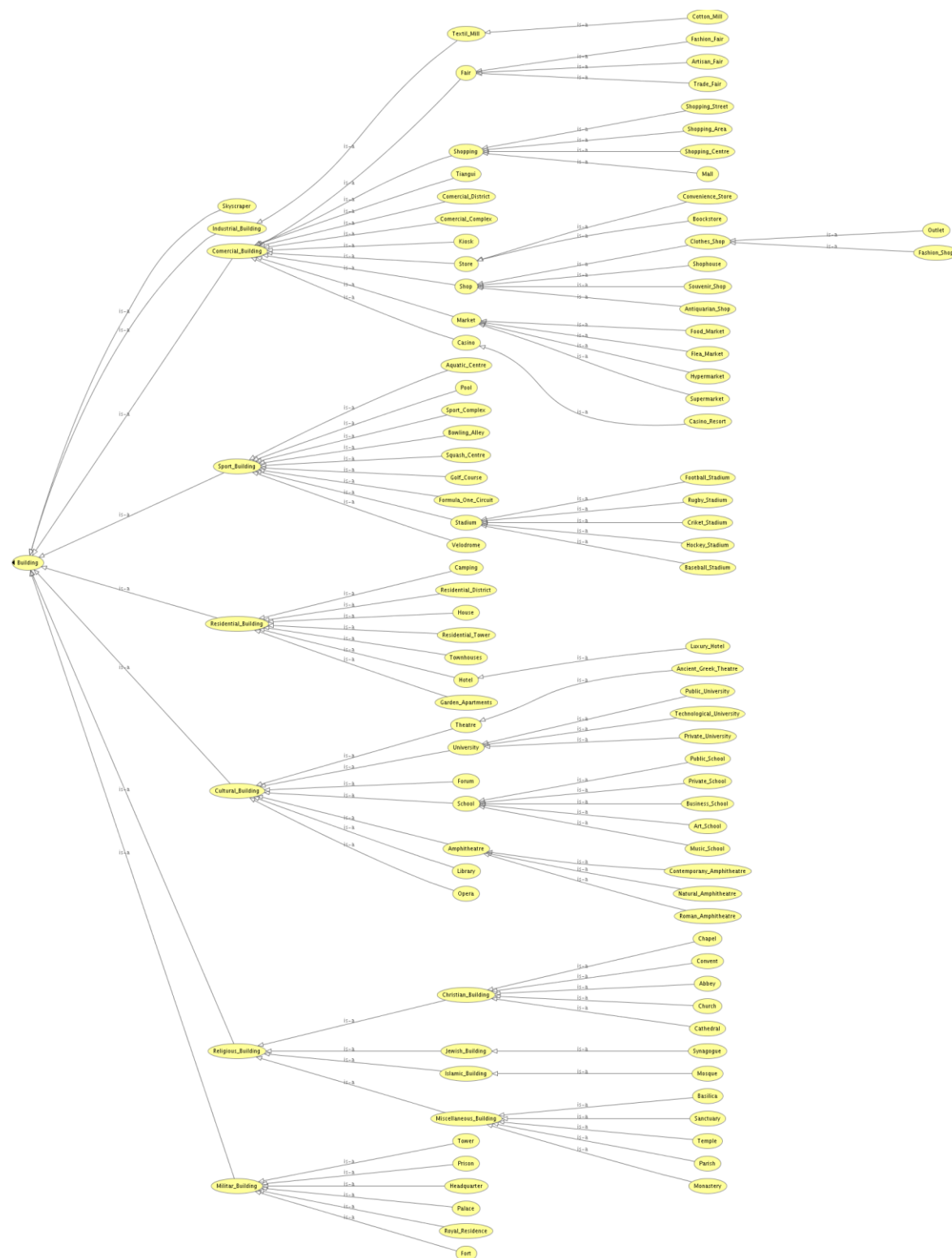


Figura 18 Taxonomia de la classe *Building*

Classe Sport

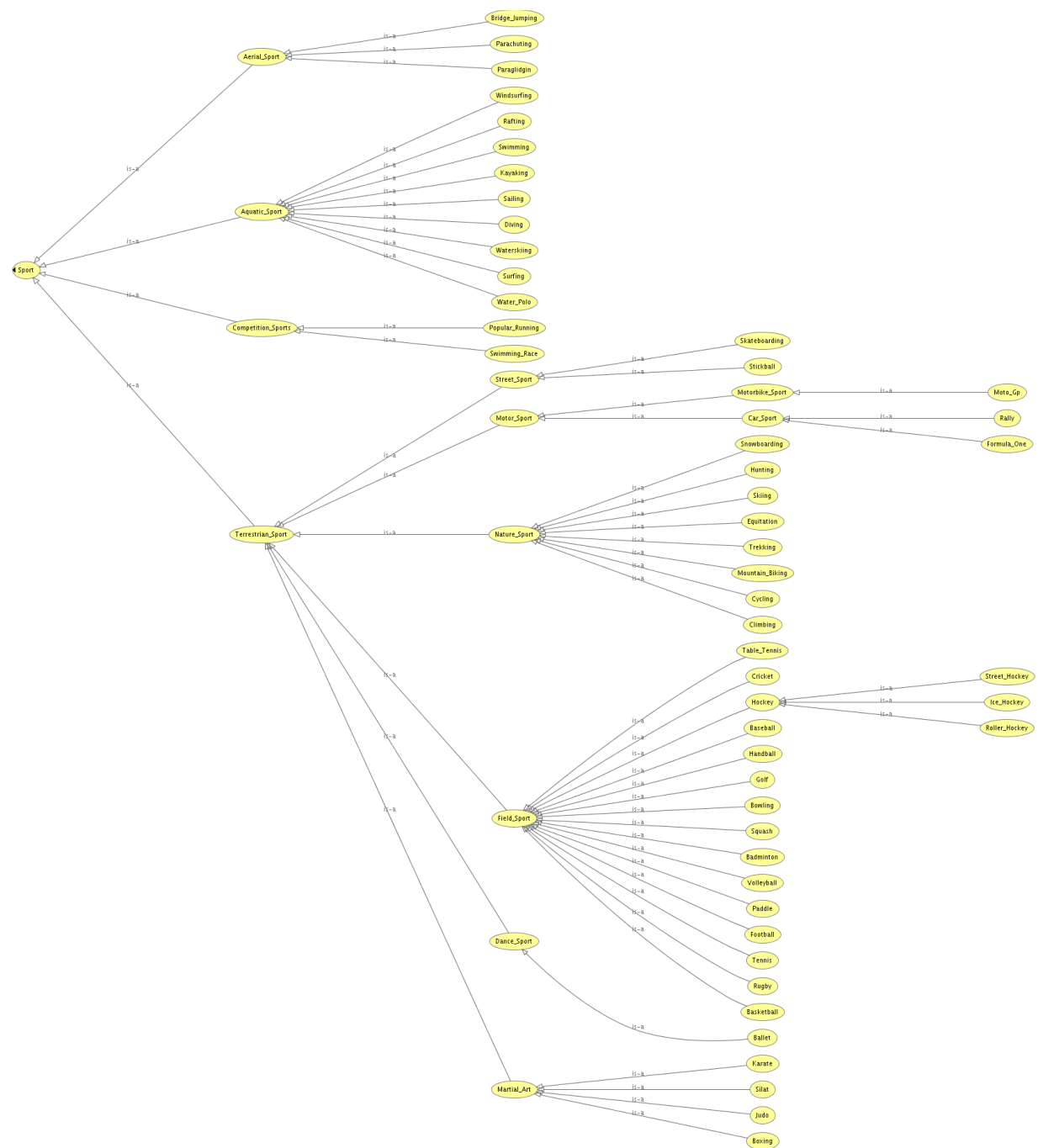


Figura 19 Taxonomia de la classe *Sport*

Classe Landmark

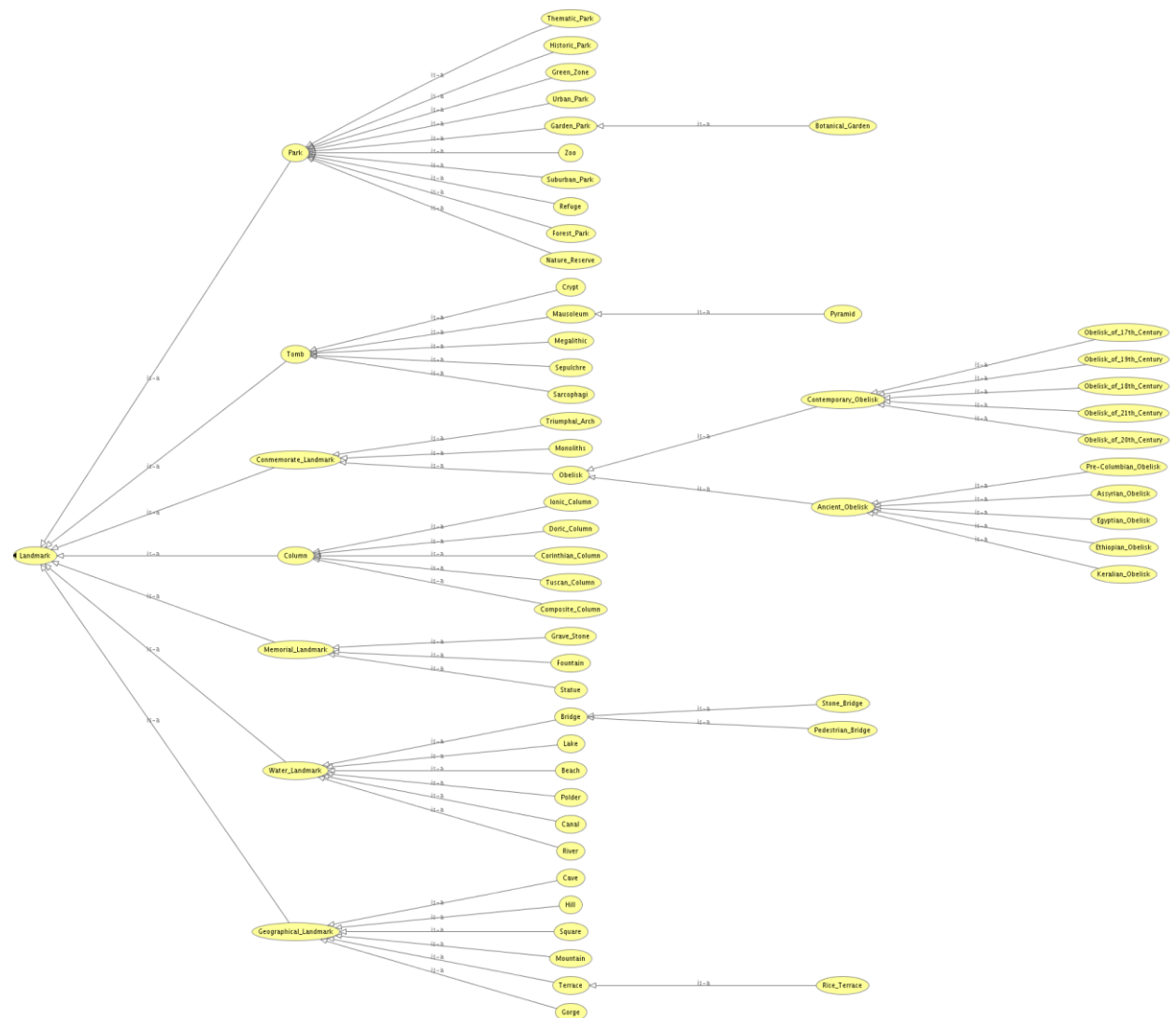


Figura 20 Taxonomia de la classe Landmark

Classe *Architecture_Styles*

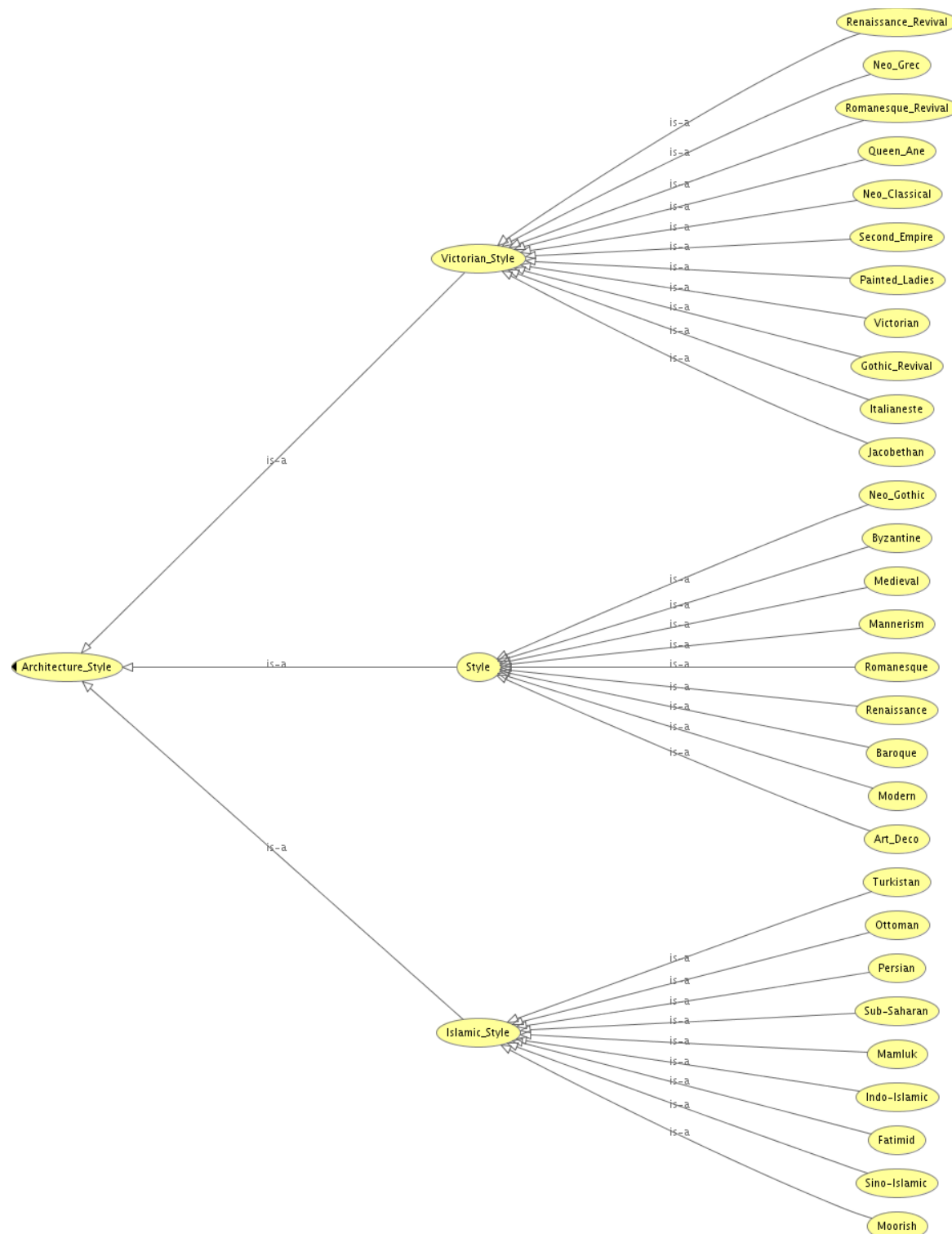


Figura 21 Taxonomia de la classe *Architecture_Styles*

Per acabar, es mostraran les relacions existents entre classes, amb l'ajuda d'un gràfic *domain/range*. Aquest gràfic és interessant per veure com es relacionen les instàncies de la classe *Building* amb les de la classe *Architecture_Styles*.

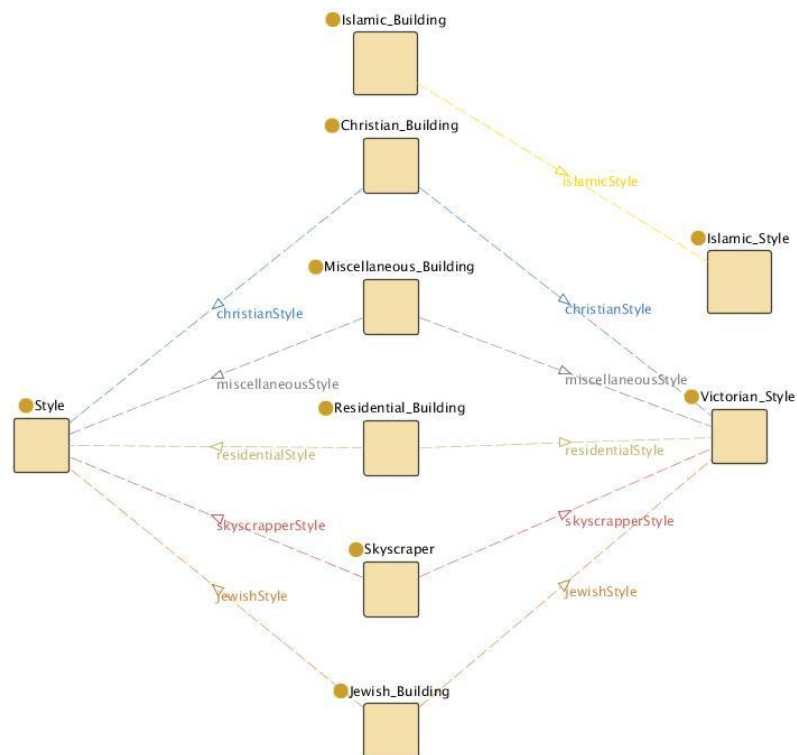


Figura 22 Relaciones *domain/range* entre las clases de l'ontologia

3 LA WIKIPEDIA

En aquest capítol es donarà una visió general sobre la Wikipedia i s'explicarà la seva estructura i característiques principals. A més, s'especificaran les parts utilitzades en aquest projecte, les eines de recuperació i *parsing* i es mostraran exemples reals sobre planes de destinacions turístiques.

3.1 Conceptes generals

La Wikipedia és una enciclopèdia multilingüe lliure mantinguda per la Fundació Wikimedia, una organització sense ànim de lucre. El seu nom està compost per un manlleu del hawaïà, *wiki*, que vol dir ràpid, i la paraula *enciclopèdia*. Va ser creada el gener del 2001 per Jimmy Wales i Larry Sanger i actualment és l'obra de referència més gran i popular a Internet, editada en 262 idiomes. Els més de 13 milions d'articles, dels quals 2,9 milions són en anglès, han estat escrits voluntàriament per usuaris d'arreu del món, i la majoria poden ser editats per qualsevol persona que pugui accedir a la Web[36].

La Wikipedia s'executa sobre el software lliure Mediawiki en un clúster de servidors Ubuntu situats a Florida i altres llocs del món.

3.2 Estructura de la Wikipedia

Per conèixer com funciona la Wikipedia i com està estructurada, mostrarem el disseny de la seva base de dades que principalment està format per 8 blocs diferenciats (Figura 23):

- Bloc 1 (*IP blocks*): manté una relació de les IP's i els usuaris sense permís per editar la Wikipedia.
- Bloc 2 (*Statistics and logging*): està format per 3 taules: *site_stats*, *logging* i *hitcounter*, i s'utilitza per generar estadístiques sobre els llocs Web i els usuaris.
- Bloc 3 (*User accounts and privileges and watchlist*): està compost per 4 taules, guarda tota la informació relacionada amb els usuaris, els seus permisos, les pàgines visitades per cada usuari i les converses entre ells.
- Bloc 4 (*Caching tables*): manté una cau sobre informació, peticions, etc. per optimitzar el rendiment de les consultes.
- Bloc 5 i 6 (*Parser testing i miscellaneous*): aquests blocs són opcionals.
- Bloc 7 (*Images and media*): guarda tota la informació relacionada amb les imatges i arxius multimèdia que es troben a la Wikipedia (imatges afegides i esborrades, noms, mides, links, etc.).
- Bloc 8 (*Article text and associated information*): És el bloc més gran de tots i està format per 13 taules: *revision*, *recentchanges*, *text*, *archive*, *templatelinks*, *categorylinks*, *pagelinks*, *pagerestrictions*, *searchindex*, *langlinks*, *externallinks*, *redirect* i *page*.

Aquest últim bloc (Figura 24) es especialment atractiu per aquest projecte i en ell centrarem la major part de la nostra atenció. Mitjançant la combinació de les metadades associades a cada article (*associated information*), la informació estructurada i el text pla crearem la nostra base del coneixement per a les destinacions turístiques.

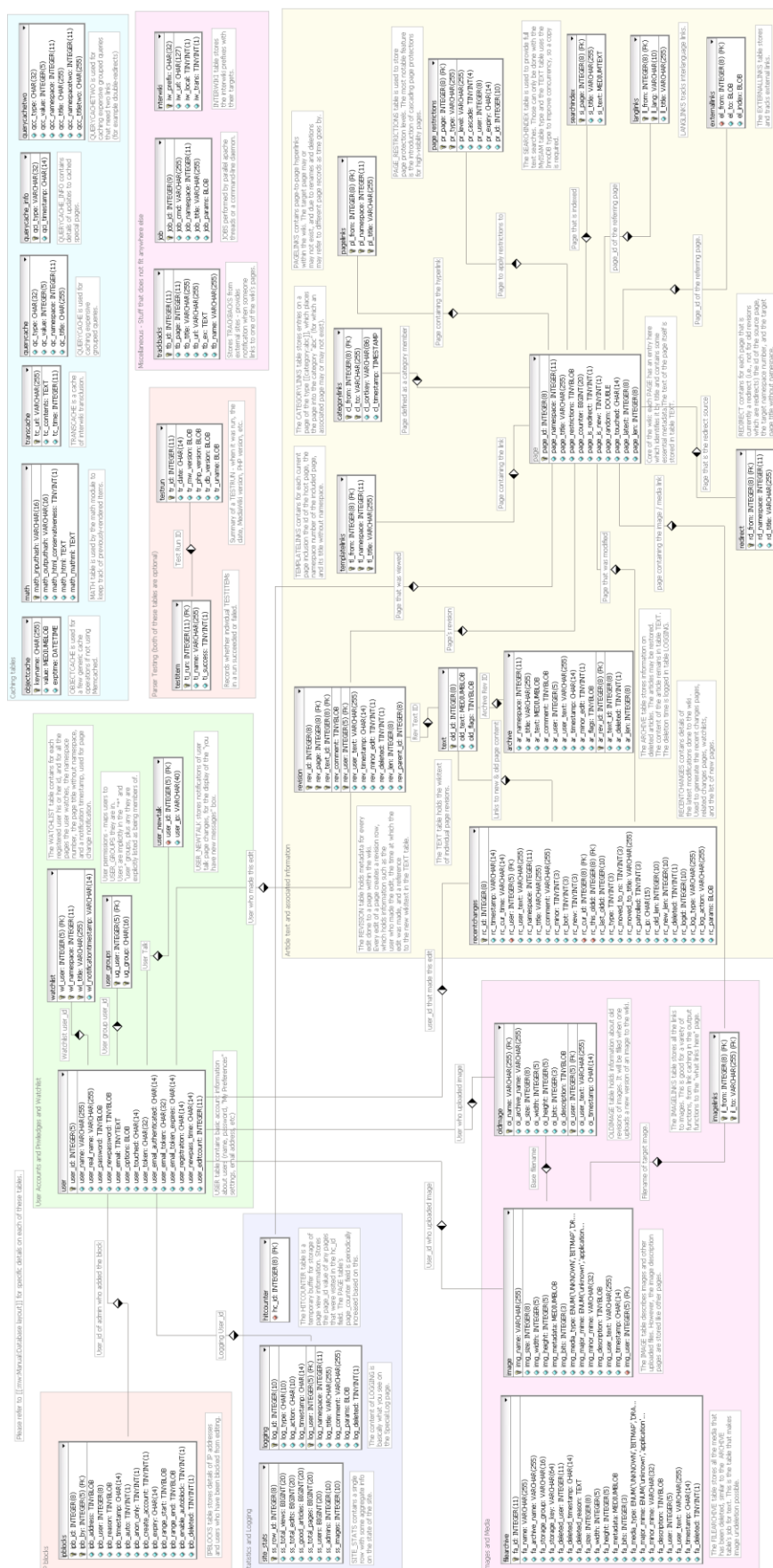


Figura 23 Esquema de la base de dades de la Wikipedia

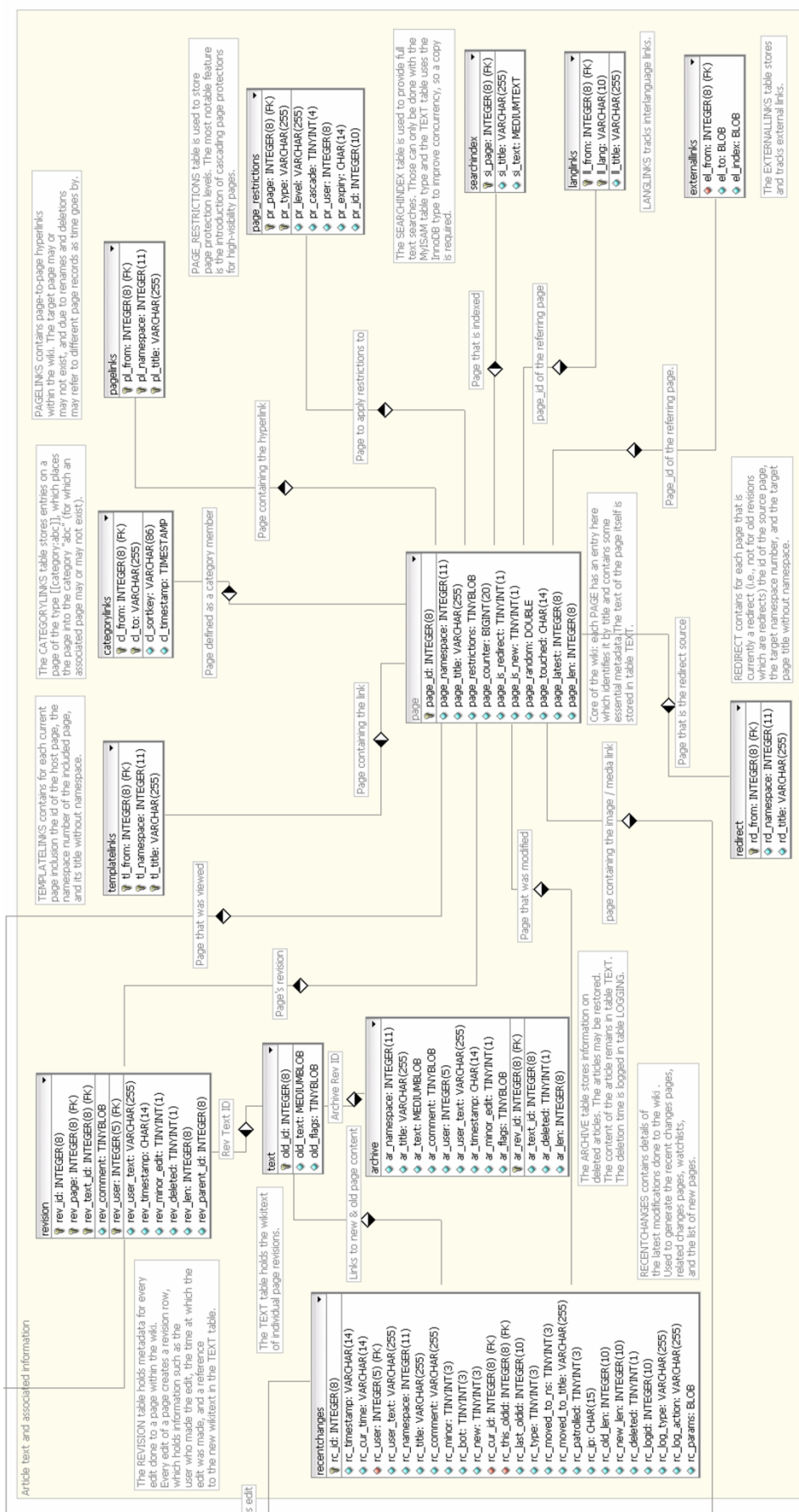


Figura 24 Esquema Bloc 8, base de dades Wikipedia

3.3 Metadades

Les metadades són un conjunt d'elements descriptius que serveixen per a la identificació de documents o recursos digitals[48].

Per a alguns camps de la informàtica, com la recuperació d'informació o la Web semàntica, les metadades en etiquetes són un enfocament important per aportar informació semàntica. L'ús de les metadades més freqüent és el refinament de consultes a buscadors, que aconsegueix resultats més precisos i estalvia feina manual de filtratge a l'usuari.

En les cerques a buscadors es planteja el problema que l'usuari i l'ordinador no s'entenen perquè aquest últim no comprèn el significat de les dades. Les metadades possibiliten aquesta comunicació declarant com estan relacionades les dades. Per aquest motiu la representació del coneixement utilitza metadades per categoritzar informacions de la mateixa manera que facilita als mètodes d'Intel·ligència Artificial la capacitat per deduir conclusions automàticament.

La Wikipedia ofereix molts tipus de metadades com per exemple el títol i URL d'un document, l'autor i moment d'edició d'una revisió, la font i llicència d'un arxiu multimèdia, el nombre d'habitants i moneda d'una ciutat, etc. D'aquesta manera, darrere el que es mostra als usuaris hi ha una xarxa de relacions i informació estructurada que aporta coneixement semàntic sobre alguns dominis.

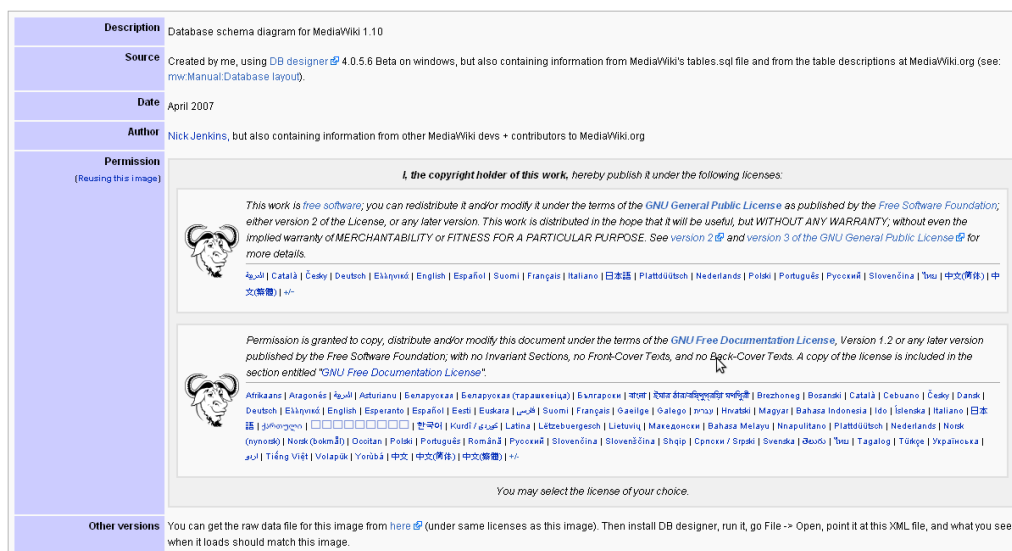


Figura 25 Exemple de metadades associades a una imatge[32]

3.4 Informació estructurada

A la Wikipedia podem trobar diferents tipus d'informació estructurada on diferenciem l'estructura dels links (Link structure) i les dades estructurades (Structured data records).

Els links aporten informació sobre relacions i connecten la part textual amb el nivell conceptual. Hi ha diversos tipus de links a destacar: els links a categories (category links) i els links interns (internal links o pagelinks), que són els que s'utilitzaran en aquest projecte. Els links interns representen una de les característiques més importants de la Wikipedia, ja que faciliten al lector la navegació entre articles relacionats. La tècnica més fàcil per crear vincles entre articles és utilitzant la sintaxis [[Títol_de_l'article]]; per exemple, a la secció "History" de la plana de Tarragona apareix un enllaç a un article que parla sobre el riu Ebre ([[Ebro]]), veure Figura 26. No obstant, utilitzant aquesta

metodologia, tots els vincles que llegeix el lector es corresponen amb els títols dels articles que és referencien. Per evitar-ho, es poden utilitzar altres tècniques com en el cas de la Figura 26, on es relaciona la paraula “Roman” amb l'article Roman_empire ([Roman_Empire|Roman])[40].

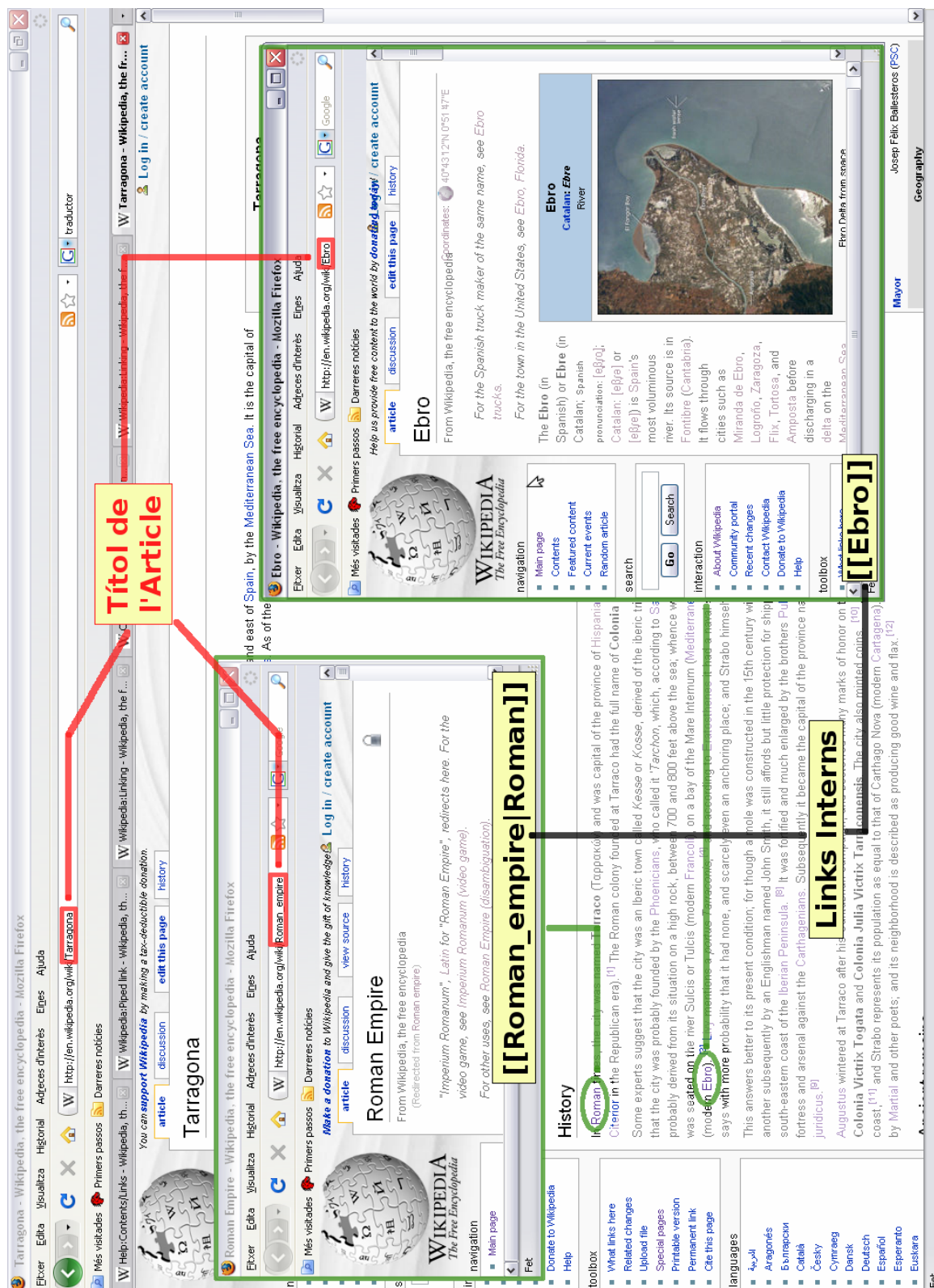


Figura 26 Exemple links interns de la Wikipedia

Els links categòrics representen categories i tenen una importància fonamental. Principalment, serveixen per a agrupar el coneixement i fer-lo més accessible. Normalment, aquesta classificació del coneixement es fa de manera jeràrquica o en forma d'arbre (per exemple, la categoria de Trobadors apareix dins de la categoria de Literatura occitana i aquesta, al seu lloc, està dins de la categoria de Literatura, que està englobada en la categoria d'Art, Art en Cultura i, finalment, Cultura en Principal). D'aquesta manera, no cal incloure categories que ja estan implícites en la jerarquia. Per exemple, la categoria *Baix Ebre* es pot relacionar amb les següents categories: *Comarques de Catalunya* -> *Geografia de Catalunya* -> *Catalunya*, etc. (cada categoria és una subcategoria de la de la que té per sobre, Figura 27). Així, en un article d'un poble del Baix Ebre, cal posar-hi que pertany a la categoria *Baix Ebre*, però no a la de *Catalunya*, perquè *Baix Ebre* es indirectament una subcategoria de *Catalunya*[37][34].

Existeixen molts altres tipus de links estructurats, tal i com es mostra en la Figura 28.

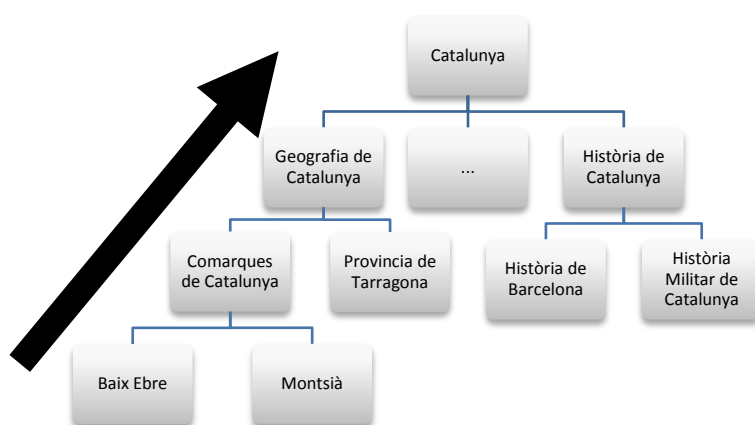


Figura 27 Categories de la Wikipedia per a Catalunya

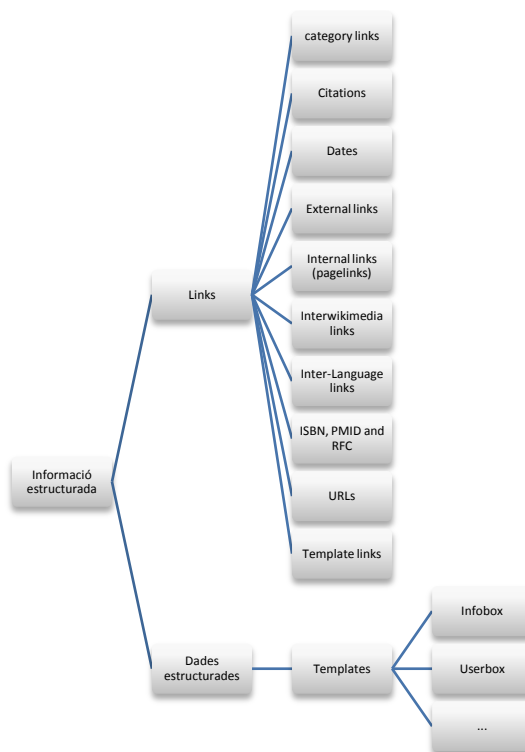


Figura 28 Classificació d'informació estructurada de la Wikipedia

3.4.1 Wikipedia Templates

Les plantilles o “Templates” són un tipus especial de pàgines de la Wikipedia i una eina important per a la representació de les dades estructurades. Totes les plantilles comencen amb el terme “*Template:*” i es categoritzen sota la categoria *Templates* o alguna de les seves subcategories. En aquestes pàgines s'hi inclouen una sèrie de codis que generen estructures de sintaxi informàtica més aviat complexa. Així doncs, les plantilles faciliten la inserció d'informació dintre d'altres pàgines. A més a més, estandarditzen i unifiquen la forma en que la informació és presentada en els articles i faciliten l'edició de les pàgines a usuaris sense grans coneixements de codificació informàtica.[31] [41]

Existeixen bàsicament tres tipus de plantilles: les plantilles sense paràmetres (*Templates with no parameters*), les plantilles amb paràmetres no identificats (*Templates with unnamed parameters*) i les plantilles amb paràmetres coneguts (*Templates with named parameters*). Combinant aquestes dos últimes es podria aconseguir un quart tipus de plantilla híbrida amb paràmetres coneguts i desconeguts[35]. Els paràmetres serveixen per representar el valor de les dades contingudes a la plantilla.

Per utilitzar cada tipus de plantilla es fan servir diferents sintaxis. En el primer dels casos, simplement s'ha d'afegir a l'article el nom de la plantilla desitjada entre claus `{{Nom de la plantilla}}`. En d'altres, la representació depèn del tipus de paràmetres (Figura 29):

- *Unnamed parameters*: els valors dels paràmetres s'afegeixen a la plantilla en l'ordre amb el que apareixen. Sintaxis: `{{ template name | parameter1 | parameter2 | ... | parameterN }}`.
- *Named parameters*: els valors dels paràmetres estan associats amb un nom clau de la plantilla. Sintaxis: `{{ template name | parametername1 = parameter1 | parametername2 = parameter2 | ... }}`.
- Combinació dels casos anteriors: `{{ template name |parameter1|parameter2|parametername1 = parameter3 |parametername2 = parameter4 | ... }}`.

Sintaxis de la Plantilla	Informació mostrada	Descripció						
<code>{{ESP}}</code>	 Spain	S'utilitza la plantilla ESP que mostra per pantalla la bandera i nom de país a la que representa.						
<code>{{green Hello World}}</code>	Hello World	S'utilitza la plantilla green amb un unnamed parameter. El que fa es mostrar el contingut del paràmetre amb color verd.						
<code>{{ diamonds 9}}</code>	9♦	Mostra la carta corresponent al valor que es passa per paràmetre.						
<code>{{payoff matrix UL = 5 UR = 7 DL = 2 DR = 9 Name = Example usage }}</code>	Left Right <table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;">Up</td><td style="padding: 2px;">5</td><td style="padding: 2px;">7</td></tr> <tr> <td style="padding: 2px;">Down</td><td style="padding: 2px;">2</td><td style="padding: 2px;">9</td></tr> </table> <i>Example usage</i>	Up	5	7	Down	2	9	En aquest cas s'utilitza la plantilla payoff matrix especificant cinc named parameters. Finalment genera i mostra una matriu amb els valors dels paràmetres especificats.
Up	5	7						
Down	2	9						

Figura 29 Exemple d'ús de plantilles

3.5 Editar la Wikipedia

Com ja s'ha comentat anteriorment, la Wikipedia és el resultat de milers d'editors d'arreu del món. Aquestes persones, anònimes o no, representen a una població molt diversa i, per tant, amb ideologies i rols ben diferenciats. Qualsevol persona amb una connexió a Internet (Advocats, metges, periodistes, enginyers, estudiants, mestresses de casa, etc.) pot editar o crear un article a la Wikipedia. És per aquest motiu que aquesta ha esdevingut una icona de referència arreu del món. Aquest fet, juntament amb la gran voluntat d'ajudar de les persones, presenta grans avantatges, però també s'hi troben alguns inconvenients. Dos de les principals crítiques que recauen sobre la enciclopèdia de moda fan referència a la fiabilitat i exactitud dels articles.

Per aquests motius la Wikipedia ha publicat una política d'edició[38], juntament amb un gran nombre de manuals d'ajuda i d'estil[39], que haurien de seguir totes les persones interessades en publicar-hi alguna cosa.

Es demana una escriptura clara, informativa i imparcial. Alguns dels consells a seguir són:

- En començar un nou article, escollir un nom adequat que el representi correctament.
- No crear articles duplicats amb noms diferents. Usar les redireccions.
- L'estil i contingut dels articles han de ser *enciclopèdics*, mantenir la neutralitat i utilitzar un llenguatge impersonal.
- Intentar que els articles relacionats semànticament mantinguin una estructura pareguda. Per exemple en el cas de les ciutats.
- Afegir enllaços interns per reforçar el contingut del article, però sense fer-ne un ús abusiu.
- Utilitzar referències externes si són necessàries.
- Agrupar els enllaços externs al final de l'article sota la capçalera *==Enllaços externs==*.
- Utilitzar la sintaxis de la Wikipedia per fer els encapçalaments.
- Utilitzar sempre unitats de mesura del Sistema Internacional.
- Per escriure nombres llargs separar els milers amb espais o comes i els decimals amb punt.
- No separar els valors numèrics dels símbols que representen la unitat amb la que es mostra.
- Afegir una breu introducció a tots els articles.
- Afegir descripcions a les imatges publicades.
- Categoritzar els articles.
- Abans de crear una categoria, comprovar que aquesta no existeixi amb un altre nom.
- Utilitzar les plantilles existents.

Aquests consells només són una mostra reduïda del contingut dels manuals de la Wikipedia i, tot i la existència d'aquestes normes “no estrictes”, no tots els usuaris les segueixen. És per aquest motiu que aquest treball, en algunes ocasions en les que pot semblar trivial aconseguir alguna informació concreta, pot resultar més difícil del que és aparentment.

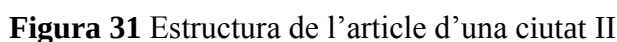
3.6 Cas particular sobre l'estructura d'un Wiki de destinacions turístiques

En aquest apartat es mostrarà, amb l'ajuda d'un exemple pràctic, l'estructura que segueixen, més o menys, els articles que s'analitzaran en aquest treball. El cas pràctic que es mostra a continuació fa referència a l'article de la ciutat de Tarragona durant el mes de Juliol de 2009.

The image shows a screenshot of the Wikipedia article for Tarragona, with various structural elements highlighted and labeled. The labels are as follows:

- Log in / create account**: Located at the top right of the page.
- Tarragona**: The main title of the article.
- INFOBOX**: Points to the summary box at the top of the article, containing a photo, a map, and a table of key facts.
- INTRODUCCIÓ**: Points to the introductory paragraph of the article.
- INDEX CONTINGUTS**: Points to the table of contents.
- Section History**: Points to the history section of the article.
- Page Link**: Points to a link in the text, likely a reference or a specific section.
- Languages Links**: Points to the multilingual link box at the bottom of the article.
- Roman Aqueduct**: Points to a specific section in the article, likely a landmark or historical site.

Figura 30 Estructura de l'article d'una ciutat I



- **Introducció:** Conté una breu introducció sobre les característiques més importats de la ciutat.
- **Índex de continguts (Table of contents):** Representa l'índex de les seccions que s'expliquen a l'article. Aquest índex es genera automàticament pel software que utilitza la Wikipedia, excepte per als articles amb menys de quatre seccions, els quals no disposaran d'índex de continguts.
- **Seccions:** Les seccions solen estar formades principalment per text i en alguns casos també contenen altres elements com imatges, taules, llistes, etc. Cada secció de l'article es troba representada com un element de l'índex de continguts.
- **InfoBox:** Es troba en la major part d'articles que parlen sobre ciutats. L'objectiu principal d'aquesta taula es mostrar un resum de les dades numèriques i categòriques més significatives sobre la ciutat en qüestió.

Sempre s'utilitza una plantilla per crear aquesta taula. El problema principal es que existeixen moltes plantilles i cap estàndard que obligui a utilitzar una plantilla genèrica per a totes les ciutats, fet que dificulta l'extracció d'informació.

- *Page Links*: Són links interns que vinculen a altres wiki articles de la mateixa llengua. Aquest tipus de links són força importants ja que ajuden al lector a augmentar la comprensió de l'article consultat.
- *Languages Links*: Serveixen per enllaçar l'article consultat amb articles d'altres llengües que tracten sobre el mateix tema.
- *External Links*: Poden haver molts motius per afegir referències a pàgines Web externes a la Wikipedia, per raons de copyright, per reforçar alguna investigació, per aportar informació extra, etc. S'utilitzen aquests links per fer-ho possible.
- *Category Links*: Representen les categories a les que pertany un article.

En aquest treball, s'intenta aprofitar aquesta semi estructura, per recuperar la informació referent a destinacions turístiques. Més endavant, s'explicaran alguns dels mecanismes i recursos que facilita el software de Mediawiki (Mediawiki Api) per tractar amb els elements esmentats anteriorment.

3.7 Recuperació i parsing (MediaWiki Api)

Per fer la recuperació i parsing de l'informació que es desitja consultar, s'utilitzarà l'API de MediaWiki[49], que proporciona accés directe a moltes de les dades contingudes a la base de dades de la Wikipedia.

L'API de MediaWiki ofereix sis mòduls principals d'entrada. Combinant aquests sis mòduls, juntament amb els seus valors i paràmetres, es poden fer consultes més o menys complexes. Els principals paràmetres són: *format*, *action*, *version*, *maxlag*, *smaxage*, *requestid*. Els resultats de sortida poden ser representats pels formats *json*, *jsonfm*, *php*, *phpfm*, *wddx*, *wddxfm*, *xml*, *xmlfm*, *yaml*, *yamlfm*, *rawfm*, *txt*, *txtfm*, *dbg* o *dbgfm*, indicant el valor del paràmetre *format*=*"format_desitjat"*, que per defecte és *xmlfm*. En aquest projecte es treballarà amb el format *xml*. Les consultes més emprades en aquest treball utilitzaran diferents opcions del modul *action*. Dos de les més destacables són:

- *action=parse*: Aquest mòdul se n'encarrega de parsejar wikitext i retornar-lo segons el format de sortida desitjat. Es poden especificar els següents paràmetres:
 - *text*: wikitext que es vol parsejar.
 - *page*: parsejar el contingut de la pàgina indicada. No es pot utilitzar juntament amb *text*.
 - *redirects*: si el paràmetre *page* activa una redirecció, la resol.
 - *prop*: indica quines parts d'informació es volen aconseguir de l'article indicat pel paràmetre *page*. Es pot escollir entre: *text*, *langlinks*, retorna la llista de *language links*, *categories*, *links*, *templates*, *images*, *externallinks*, *sections*, *revid*, *displaytitle*. En el cas de *sections*, l'API retorna l'índex de continguts de l'article, no el contingut de cada secció.
- *action=expandtemplates*: expandeix les plantilles que estan en wikitext. S'hi poden especificar tres paràmetres diferents: *title*, *text* i *generatexml*.

Al final d'aquest document s'annexa el manual complet de l'API de MediaWiki(annex 8.1)

4 DISSENY DEL SISTEMA DESENVOLUPAT

Aquest capítol està format per tres seccions principals. En la primera (secció 4.1) s'explicarà el disseny i funcionalitat de l'eina desenvolupada de tractament de matriu. En la següent (secció 4.2) es parlarà sobre el sistema analitzador implementat. Finalment, en la secció 4.3, es comentaran les limitacions generals de tots dos aplicatius.

4.1 Eina tractament de matriu

L'objectiu principal d'aquest projecte és la creació d'una matriu atribut-valor amb informació de destinacions turístiques. Es vol, però, que aquesta matriu sigui el més flexible possible respecte a la informació que es vol buscar. És per aquest motiu que s'ha decidit implementar una eina que permeti crear i manegar aquesta matriu.

Com s'ha explicat a l'apartat 1.4.3 del capítol 1, aquesta eina disposa de dos modes de funcionament. El primer d'ells permet generar una matriu buida, on cada columna representa la informació que es vol buscar i cada fila, una destinació turística diferent, de manera que en qualsevol moment es pot crear una matriu que contingui informació més específica o més general respecte al contingut que es vol aconseguir (Figura 4). Existeixen tres casos d'informació que es poden representar en aquestes columnes. En el primer, la columna pot representar una classe de l'ontologia que té fills, per exemple *Religious_Building*, aleshores els valors continguts en aquesta columna seran les seves subclasses (*Cathedral*, *Church*...). El segon simbolitza una classe de l'ontologia que no té fills, per exemple *Football*, en aquest cas el contingut de la matriu només podrà ser el conjunt buit '?' o la classe *Football*. L'últim dels casos representa atributs d'alguna de les classes com per exemple el nombre d'habitants.

El segon mode de funcionament comprén alguns mecanismes encarregats de convertir una matriu donada a diferents formats amb l'objectiu de poder ser analitzada o tractada per algun aplicatiu extern (Figura 5).

En els següents apartats es mostra una visió general de l'estructura de l'aplicatiu, els seus algorismes principals i el seu funcionament.

4.1.1 UML

En aquesta secció es mostraran i explicaran els diagrames de modelatge de l'aplicació, la seva estructura i la finalitat de cada classe o paquet que haurà de ser creat.

L'aplicatiu estarà format per tres paquets principals: *application*, *interface* i *tree* i per dos paquets auxiliars: *test* i *wrapper* (Figura 32). Aquest últim s'utilitza com a envoltori de l'ontologia i facilita els mètodes necessaris per manegar-la.

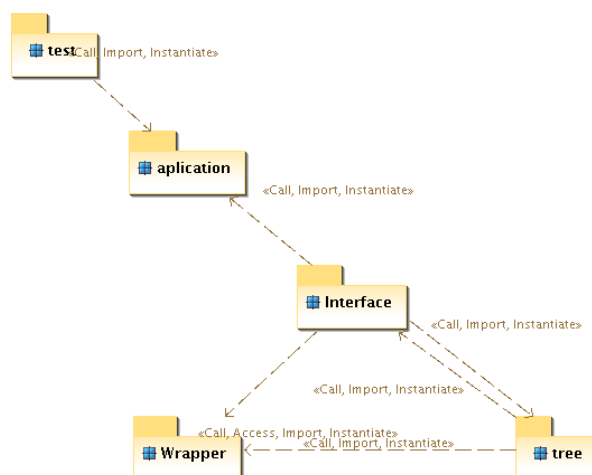


Figura 32 Dependències entre els paquets de l'eina de tractament de matriu

4.1.1.1 Application

Aquest primer paquet contindrà els tipus abstractes de dades, les classes encarregades de representar la matriu i els mètodes per gestionar-la. Està format per les classes *CreatorOutputFormat*, *Matrix* i *Dialogs*.

La classe *CreatorOuputFormat* és l'encarregada de proporcionar tots els mètodes necessaris per convertir una matriu d'un format a un altre. Es contemplen conversions entre els formats *.properties*, *.ser*, *.xls* i *.klass*. En la Figura 33 es mostren tots els mètodes de conversió que facilita.

La classe *Matrix* representa a les dades del tipus Matriu atribut-valor. Aquesta classe s'utilitzarà tant en l'aplicatiu de tractament de matriu, com en l'aplicatiu analitzador de destinacions turístiques. Aquesta classe implementa la interfície serialitzable i, mitjançant els mètodes *saveMatrix* i *loadMatrix*, es pot guardar qualsevol instància seva a un fitxer extern i carregar-lo des d'una altra aplicació. En la Figura 33 es mostren tots els seus mètodes.

Finalment, la classe *Dialogs* s'utilitza per centralitzar tots els missatges d'informació i error que es mostraran a l'usuari de l'aplicatiu. D'aquesta manera s'aconsegueix una gran transparència en la programació.

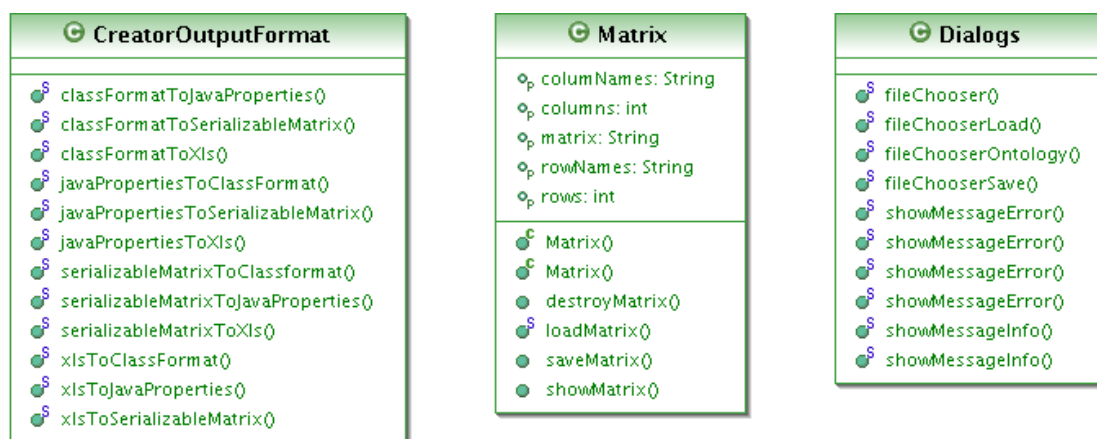


Figura 33 Classes i mètodes del paquet *application*

4.1.1.2 Interface

El paquet *interface* conté tots els elements i events necessaris per gestionar la interfície gràfica de l'aplicatiu. Està format per les classes *MyInterface*, *MyList*, *MyMenuBar*, *MenuBarEvent* i *MenuBarEventListener*.

Totes aquestes classes hereten o implement algun element de *java swing* o *java awt* adaptant el seu funcionament a les necessitats de l'aplicatiu.

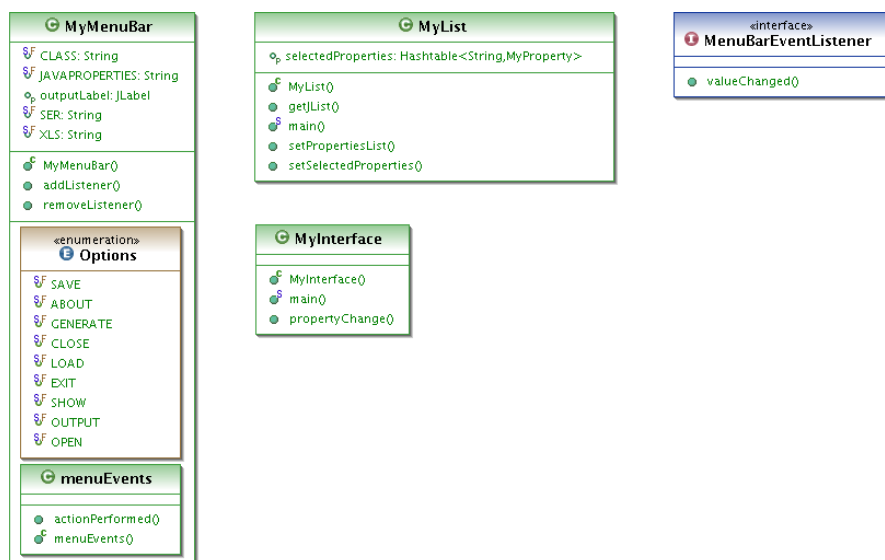


Figura 34 Classes i mètodes del paquet *interface*

La classe *MyList* s'utilitza per gestionar i personalitzar els elements d'una llista mitjançant una instància de la classe *JList*. Els elements que s'emmagatzemen en aquesta llista són del tipus *MyProperty*. Aquests elements poden representar propietats del tipus *object properties* o *data type properties* d'una classe de l'ontologia. La característica principal d'aquesta classe és que pot retornar una llista amb totes les propietats seleccionades per l'usuari. És important observar que la classe *MyList* s'encarrega de pintar-se per pantalla a ella mateixa i de tractar els events que es generen quan l'usuari interactua amb ella.

La classe *MyMenuBar* hereta de la classe *JMenuBar* i representa una barra de menú personalitzada per a l'aplicatiu. Aquesta barra es gestiona igual que la classe anterior.

La classe *MyInterface* hereta de la classe *JFrame* i esdevé la classe principal. Aquesta agrupa tots els elements de l'aplicatiu i se n'encarrega de la interacció i gestió general amb l'usuari, així com de l'execució del programa principal. En la següent figura (Figura 35) es mostren les relacions i associacions d'aquesta amb tots els elements i classes de l'aplicatiu.

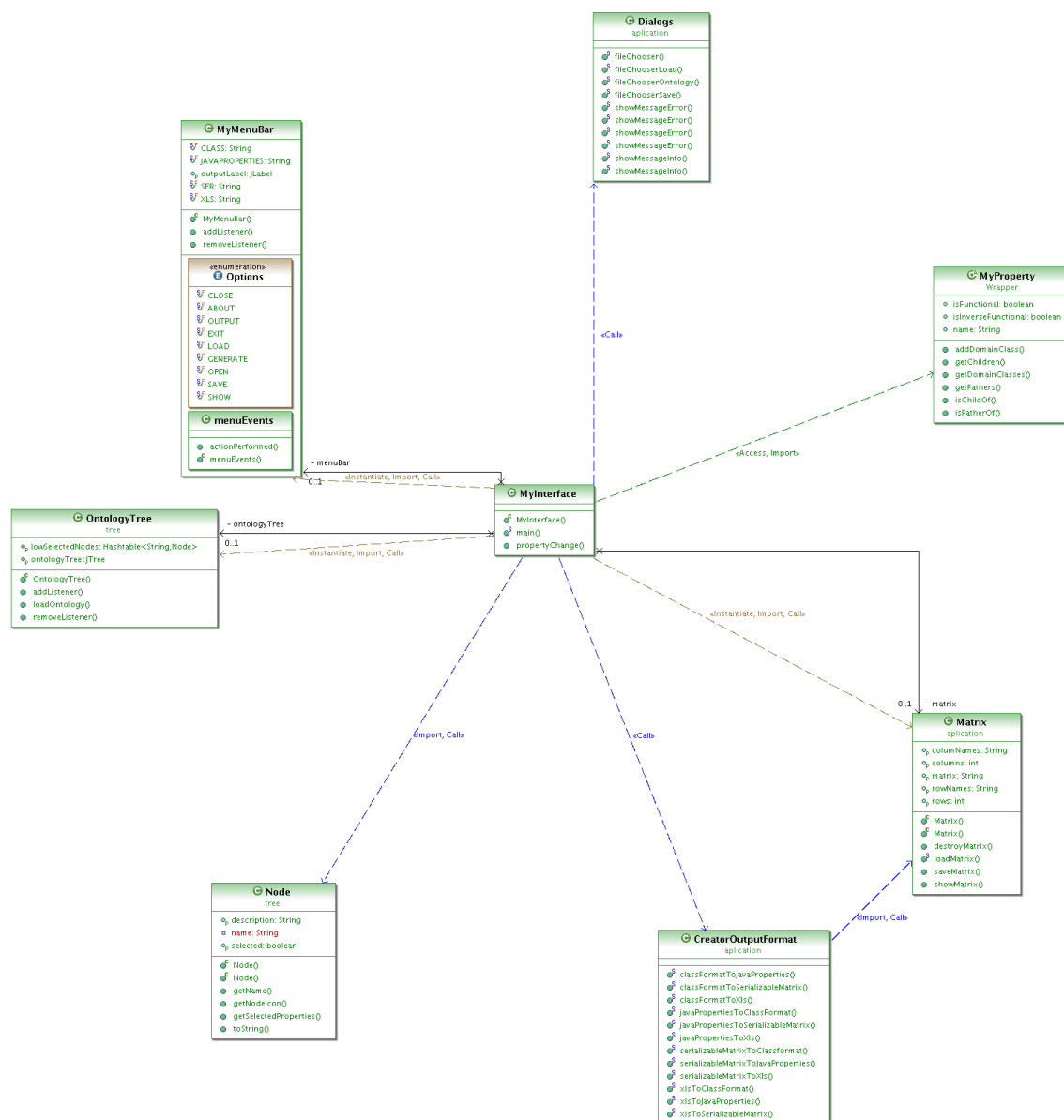


Figura 35 Associacions i dependències de la classe *MyInterface*

4.1.1.3 Tree

Finalment, el paquet *tree* conté les classes necessàries per representar, administrar i mostrar l'ontologia en forma d'arbre. Està format per les classes *OntologyTree*, *Node*, *TreeEvent* i *TreeEventListener*. Aquestes dos últimes s'utilitzen per representar els events generats per l'usuari i la classe *OntologyTree*. En la Figura 36 es mostren les classes i els mètodes de cadascuna.

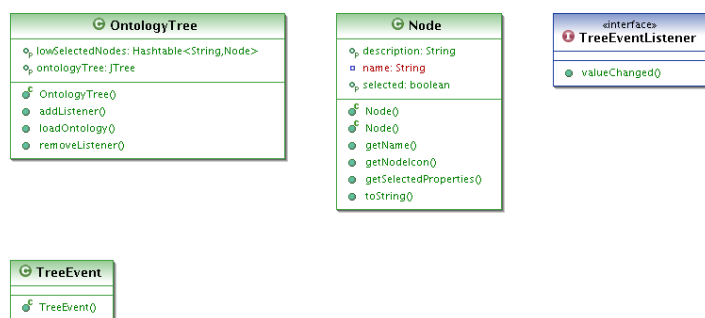


Figura 36 Classes i mètodes del paquet *tree*

La classe *Node* simbolitza un element de l'arbre. Concretament, representa una classe de l'ontologia (*MyClass*). A més a més, cada node conté un atribut per identificar si l'usuari l'ha seleccionat o no i mètodes per personalitzar les icones amb les quals es mostra en funció del valor de la selecció.

Per acabar, la classe *OntologyTree* personalitza els events generats per l'usuari i emmagatzema l'ontologia que serà mostrada en forma d'arbre. A més, ofereix mètodes per retornar una llista amb tots els *Nodes* seleccionats. A continuació es poden veure les relacions d'aquesta classe amb totes les altres de l'aplicatiu (Figura 37), i en la Figura 38 es mostra una finestra dividida verticalment en dos parts on s'hi pot veure la representació gràfica que genera la classe *OntologyTree* i els seus *Nodes* al costat dret i la representació gràfica de la classe *MyList* del paquet anterior al costat esquerre.

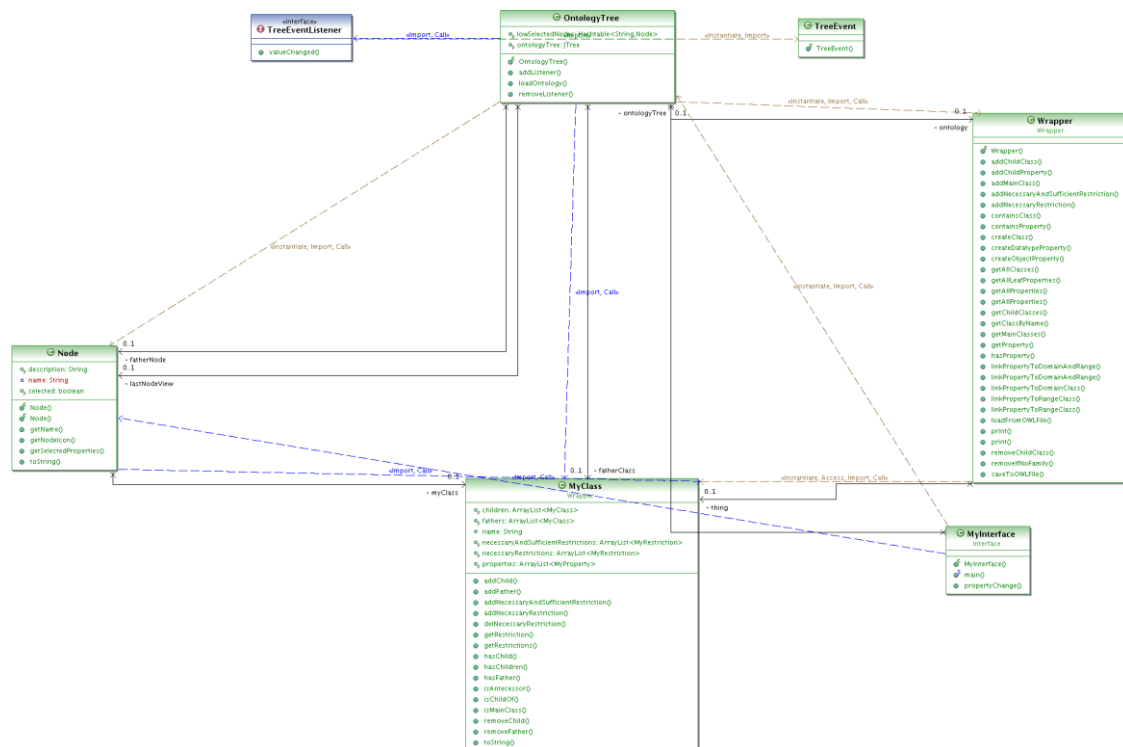


Figura 37 Associacions i dependències de la classe *OntologyTree*

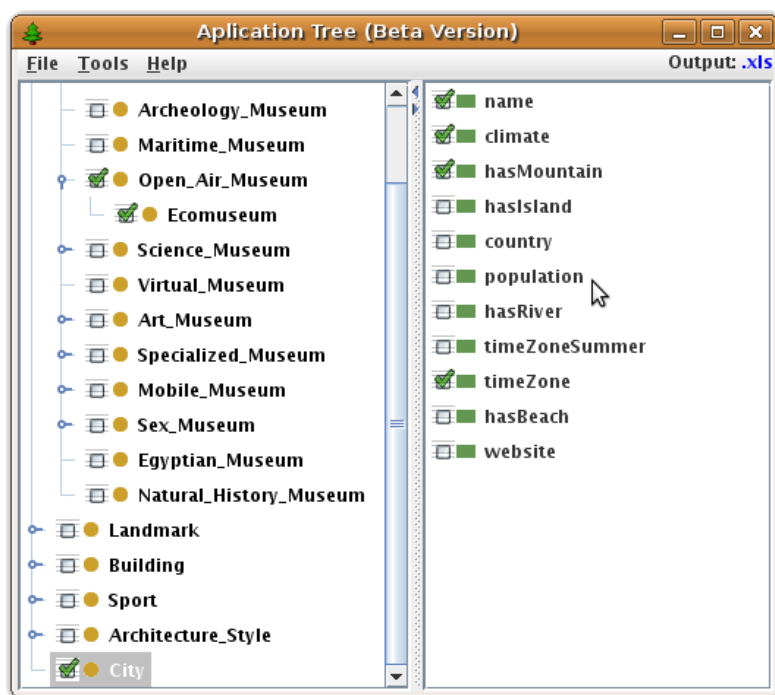


Figura 38 Representació gràfica d'*OntologyTree* i *MyList*

4.1.2 Selecció i des-selecció de nodes

L'algoritme més característic d'aquesta eina és el que se n'encarrega de seleccionar o des-seleccionar les classes (o nodes) escollides per l'usuari. Existeixen dos casos possibles: quan l'usuari selecciona un node, l'algoritme selecciona recursivament a tots els seus antecessors; en canvi, quan l'usuari el des-selecciona, l'algoritme actua de manera inversa recorrent recursivament tots els seus successors amb la finalitat de des-seleccionar-los tots.

```
private void recursiveSelection(DefaultMutableTreeNode node,
                              Boolean select){
    if (node!=null) {
        Node myNode = getNodeFromDefaultMutableTreeNode(node);
        if (select) {
            selectNode(myNode, select);
            lowSelectedNodes.remove(myNode.getName());
            recursiveSelection((DefaultMutableTreeNode)node.getParent(),
                              myNode.isSelected());
        }
        else{
            for (Enumeration<DefaultMutableTreeNode> e = node.children();
                 e.hasMoreElements();) {
                DefaultMutableTreeNode nodeAux = e.nextElement();
                Object nodeInfoaux = nodeAux.getUserObject();
                Node myNodeaux = (Node)nodeInfoaux;

                if (myNodeaux.isSelected()) {
                    myNodeaux.setSelected(select);
                    ontologyTree.repaint();
                    lowSelectedNodes.remove(myNodeaux.getName());
                    if (!nodeAux.isLeaf())
                        recursiveSelection(nodeAux, select);
                }
            }
        }
    }
}
```

4.1.3 Funcionament

En aquest apartat s'explica la utilitat dels botons de la barra de menús de l'aplicatiu.

El primer és el menú *File*; en ell s'hi troben els botons *Open Ontology*, *Load Matrix*, *Close Ontology* i *Exit* (Figura 39).

- *Open Ontology*: Amb aquesta opció es carrega i mostra l'ontologia escollida per l'usuari en forma d'arbre (Figura 38) i s'inicia el mode de funcionament A (Figura 4).
- *Load Matrix*: S'utilitza per carregar i mostrar una matriu existent. L'objectiu d'aquesta opció es posar en marxa el mode de funcionament B (Figura 5) per poder visualitzar o realitzar algún tipus de conversió sobre la matriu carregada.
- *Close Ontology*: Descarrega l'ontologia i finalitza el mode de funcionament A.
- *Exit*: Surt de l'aplicatiu.

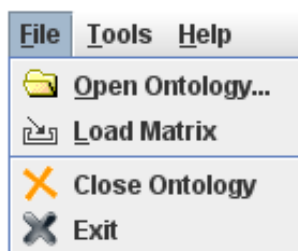


Figura 39 Menu *File* de l'eina de tractament de matriu

El següent menú és el menú d'eines (*Tools*). En aquest, es troben les opcions relacionades amb la manipulació de la matriu (Figura 40).

- *Generate matrix*: Actúa de manera diferent segons el mode de funcionament activat. En el mode A, es genera una matriu buida amb les classes i atributs triats per l'usuari. De totes les classes escollides només s'afegeixen aquelles que no tenen cap classe filla seleccionada, o sigui, les classes elegides de nivell més baix. En el mode B, es converteix una matriu d'entrada al format especificat a *Output*.
- *Show matrix*: Si existeix alguna matriu carregada, la mostra per pantalla.
- *Save matrix*: Guarda la matriu en un fitxer extern amb el format especificat per *Output*.
- *Output*: Especifica el format de sortida de la matriu. Es pot escollir entre els formats *.ser*, *.klass*, *.xls* i *.properties*.

L'últim menú (menú *Help*) conté una única opció que mostra informació sobre l'aplicatiu, versió, etc. (Figura 41).

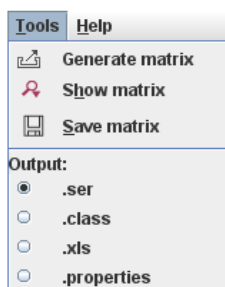


Figura 40 Menu *Tools* de l'eina de tractament de matriu



Figura 41 Menu *Help* de l'eina de tractament de matriu

4.2 Sistema analitzador

El sistema analitzador és l'encarregat d'analitzar els recursos Web i extreure tota la informació referent a destinacions turístiques. El principi de funcionament ja s'ha mostrat en el primer capítol d'aquest document (apartat 1.4.4 i Figura 6): el sistema genera una matriu atribut-valor de sortida amb totes les dades trobades, guiant-se durant el procés de cerca d'informació mitjançant una ontologia de domini.

Per donar una visió tècnica sobre el disseny i funcionament del sistema aquest apartat s'ha dividit en cinc parts; en la primera es vol donar una visió general del sistema, com estarà estructurat i quins seran tots els elements que intervindran durant el procés d'extracció i, en la resta d'apartats, s'explicaran detalladament cadascun dels elements que el formen.

4.2.1 Visió general

Donat que aquest sistema forma part d'un projecte més gran, s'ha volgut aconseguir la màxima transparència possible en la programació i, especialment, en la reutilitzabilitat del sistema que, inicialment, només extreurà informació de recursos Web semi-estructurats a partir de la Wikipedia, però, dintre del marc del projecte DAMASK, s'espera poder obtenir informació a partir d'altres recursos Web, ja siguin semi-estructurats (com en el cas de Wikitravel), estructurats (com, per exemple, alguna base de dades) o no estructurats (blogs, text pla, etc.). Per aconseguir-ho s'ha implementat el sistema basant-se en el patró de disseny *Observer*. Per veure en què consisteix aquest patró, primer s'explicarà com està estructurat el sistema analitzador i després es veurà com s'aplica en aquest projecte.

El sistema està estructurat en vuit paquets. Aquests paquets són *analizadors*, *reader*, *main*, *utils*, *exceptions*, *tads*, *Wrapper* i *test*. Els quatre primers constitueixen la part més importat de l'aplicatiu, es parlarà d'ells en un apartat dedicat a cadascun. La resta de paquets ofereixen les excepcions, alguns tipus abstractes de dades, l'envoltori de l'ontologia i varis jocs de proves.

En el paquet *analizadors* es defineix una mena de contracte que hauran de complir tots els objectes analitzadors mitjançant la "interface" *IAnalizadors*. Segons aquest "contracte", aquests hauran d'implementar una sèrie de mètodes comuns a tots ells. Aquests analitzadors s'anomenen *observadors*. Posteriorment, la classe encarregada de posar en marxa el procés d'anàlisi, anomenada *Subjecte*, mantindrà una llista d'objectes *observadors* (del tipus analitzador) i proporcionarà una interfície per afegir-los o eliminar-los. D'aquesta manera, si en un moment donat és necessari obtenir informació de recursos Web diferents, només caldrà crear un nou tipus d'analitzador i afegir-lo a la llista, sense tenir la necessitat de modificar el funcionament del sistema.

4.2.2 Analitzadors (package analyzer)

Aquest paquet està format per la *interface* *IAnalizator*, la classe abstracta *Analizator* i tots els analitzadors que s'utilitzaran en aquest projecte: *AnalizatorText*, *AnalizatorCategories*, *AnalizatorImages*, *AnalizatorInfoBox*.

Durant aquest apartat, ens referirem al nombre de conceptes trobats per l'analitzador com a coincidències trobades, i, com a coincidències afegides o simplement coincidències, al total de conceptes afegits a la matriu respecte el total de coincidències trobades.

En la següent figura (Figura 42) es pot veure com la classe abstracta implementa la *interface IAnalizator* i com totes les classes que hereten d'ella implementen els mètodes "contractats". Aquests mètodes són:

- *extract*: inicia el procés d'extracció. Conforme l'algoritme troba les dades buscades les afegeix a la matriu final. A més, quan finalitza el procés retorna el nombre de coincidències trobades i afegides a la matriu.
- *getSuccesCoincidencies*: retorna el nombre de coincidències trobades i afegides a la matriu.
- *getTotalCoincidencies*: retorna el nombre total de coincidències trobades durant l'extracció.



Figura 42 Herència de classes del paquet *analizators*

En la Figura 43 es poden veure amb més detall els mètodes de la classe *IAnalizator* amb els paràmetres i tipus de dades que retornen. El mètode més importat és el mètode *extract*, que rep 5 paràmetres: El primer d'ells és *wikiReader*, que s'utilitza per llegir la informació de la Wikipedia (veure apartat 4.2.3), els tres paràmetres següents es tracten d'una *taula de hash* formada per una paraula clau del tipus *String* i una llista de valors del mateix tipus per cada clau. El primer d'aquests paràmetres és el *synonymousCategories*, que guarda una llista de paraules que representen categories sinònimes (veure 4.2.2.2) on la

clau representa una categoria. El segon dels paràmetres és *trainedFinderSections*; el principi de funcionament d'aquest és el mateix que el de la taula anterior, a diferència que aquest conté per a cada concepte general de l'ontologia una llista de seccions possibles de la Wikipedia on possiblement es pugui trobar aquest concepte (veure 4.2.2.1). L'última d'aquestes taules és *synonymousInfoBox* i s'utilitza igual que *synonymousCategories* però, en comptes de guardar sinònims per categoria, emmagatzema sinònims per atribut de la *InfoBox*. Els dos últims paràmetres, *row* i *column*, indiquen la fila i columna a partir de la qual es poden buscar els conceptes. A més a més, en aquesta figura, també s'hi pot observar que la classe abstracta *Analizator* ofereix alguns mètodes protegits per a que les classes que l'hereten puguin utilitzar-los sense necessitat de implementar-los de nou.

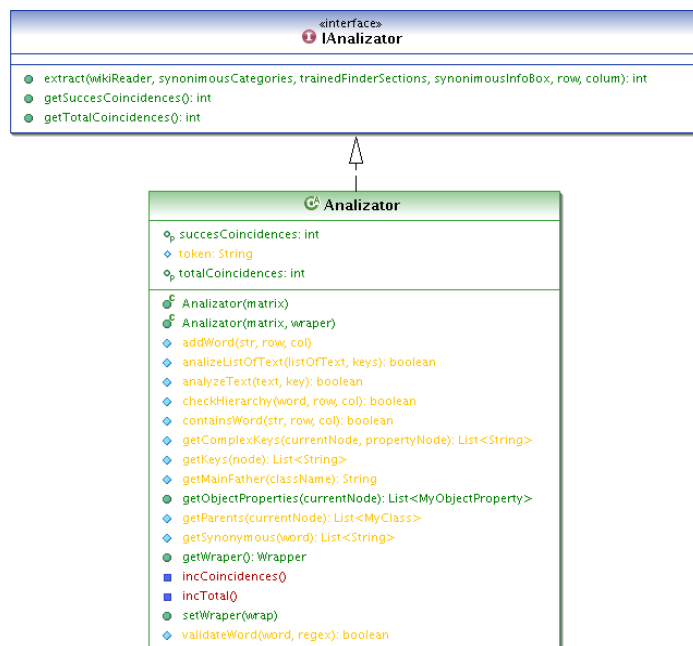


Figura 43 Mètodes públics, protegits i privats de la classe *IAnalizator* i *Analizator*

S'han creat quatre analitzadors diferents que s'encarregaran d'extreure la informació buscada en diferents parts de la Wikipedia -en els apartats 4.2.2.1, 4.2.2.2, 4.2.2.3 i 4.2.2.4 s'explica cadascun d'ells. Per fer-ho, es mostraran els principals algorismes que segueix cadascun dels analitzadors representats com a diagrames de flux. Els tres primers analitzadors funcionen amb els mateixos principis i els seus algorismes són molt semblans entre ells. A continuació, es mostra la llegenda dels diagrames (Figura 44):

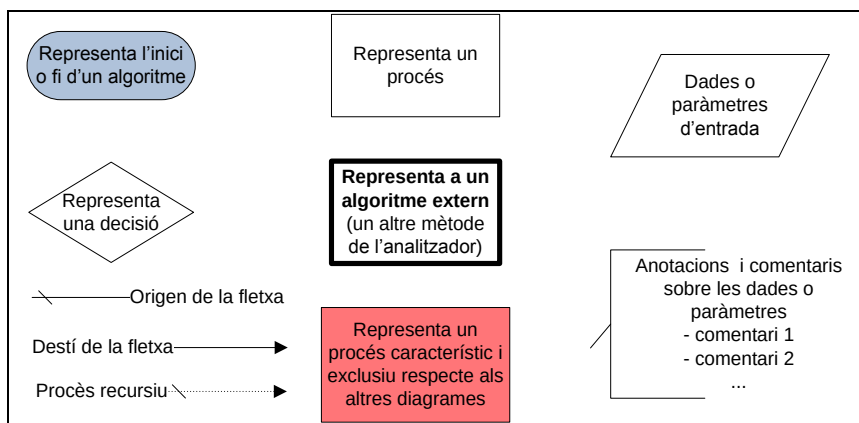


Figura 44 Llegenda diagrames de flux

Aquests tres primers analitzadors tenen en comú els mètodes *extract* (imposat per la interface *IAnalizator*), *recursiveAnalysis*, *analyzeNode*, *analyzeWithProperties* i *recursiveAnalysisProperties*. Tots aquests mètodes es comporten de manera semblant:

- *extract*: es posa en marxa el procés d'extracció. Es recorre la matriu de dades per columnes i, per a cadascuna, es decideix en quines parts de la Wikipedia es pot trobar la informació representada per la columna. S'ha de tenir en compte que cada columna pot contenir atributs numèrics o categòrics. Els primers només s'analitzaran al *AnalizatorInfoBox*. Els segons representen a una classe de l'ontologia i, per tant, a tots els elements fills d'aquesta. Llavors, per decidir quines parts s'analitzaran, s'agafa la classe pare principal de la classe representada per la columna i, mitjançant les *trainedFinderSections*, es decideix on buscar. Finalment, es posa en marxa el procés d'anàlisi recursiu.
- *recursiveAnalysis*: S'executa un procés d'anàlisi recursiu. L'objectiu és analitzar la classe representada per la columna de la matriu però recorrent l'arbre dels fills. Per tant, sempre es busca la informació de manera que es cerquen els conceptes de més específics a més generals. Si la classe fulla (classe que no té fills) no retorna coincidències, llavors s'estudia la classe immediatament superior en dos passos: primer s'analitza el node (*analyzeNode*) i després, les propietats (*analyzeWithProperties*). Si finalment no s'ha trobat cap coincidència s'estudia la llista de texts (*analyzeListOfTexts*). En el següent mètode s'explica què és aquesta llista i en què consisteix.
- *analyzeNode*: donada una classe, l'objectiu principal d'aquest mètode es tindre una llista completa amb tots els texts o elements a examinar. Per fer-ho s'aconsegueix una llista de totes les claus de *trainedFinderSections* i es comproba si alguna de les claus representa a la classe donada o a algun dels seus antecessors. En cas afirmatiu, s'obté la llista de valors (seccions de la Wikipedia) emmagatzemats per la clau i, si algun d'aquests valors es troba dintre de l'índex de la Wikipedia de la ciutat analitzada, s'afegeix el seu contingut a la llista de texts. Finalment, es retorna la llista de texts.
- *analyzeWithProperties*: Aquest mètode comproba si la classe passada per paràmetre conté alguna propietat del tipus *objectProperties*. En cas afirmatiu, s'obtenen tots els rangs d'aquesta propietat, és a dir, totes les classes que pot prendre per valor, i per a cada rang s'inicia el procés d'anàlisi recursiu amb propietats (*recursiveAnalysisProperties*). Finalment, es retorna "cert" o "fals" en funció de si s'han trobat coincidències o no.
- *recursiveAnalysisProperties*: Aquest mètode es comporta igual que *recursiveAnalysis* però es diferencia d'ell en el fet que la paraula buscada al text és una paraula composta per la concatenació de la classe que representa la propietat i la classe buscada.

S'han de tenir en compte algunes observacions importants:

- El procés d'anàlisi es basa en la comparació de Strings ("matching").
- Totes les classes de l'ontologia estan etiquetades amb, al menys, un sinònim.
- Durant el procés d'anàlisi es busquen al text analitzat totes les etiquetes pertanyents a la classe cercada.

4.2.2.1 Analitzador de text (*AnalizatorText*)

Aquest analitzador és l'encarregat d'extreure informació a partir del text de la Wikipedia. En l'exemple de la Figura 45 es mostra un cas particular en el que es busquen edificis culturals (classe *Cultural_Buildings*, veure Figura 18): en la imatge es remarca en blau la URL de l'article de la Wikipedia, en gris, el nom de la secció que està essent analitzada i, en roig, la coincidència trobada. En un moment donat, s'està recorrent l'ontologia de fills a pares en busca del mot *roman_amphitheatre*, que no apareix al text; llavors, es continua buscant als seus mots germans com *natural_amphitheatre* i *contemporary_amphitheatre*. Com no s'ha generat cap resultat, es repeteix el procediment, però aquesta vegada a un nivell més alt de l'ontologia, concretament es busca la classe *amphitheatre*, que en aquest cas retorna un resultat positiu i, per tant, s'afegeix la coincidència a la matriu de dades.



Figura 45 Exemple del procés d'extracció d'*AnalizatorText*

En aquesta figura (Figura 46) es mostra el diagrama de flux del mètode *extract* d'aquest analitzador. Es diferencia dels altres en que a la llista de texts per analitzar s'afegeix la introducció i el text de les seccions (veure Figura 30) de la Wikipèdia.

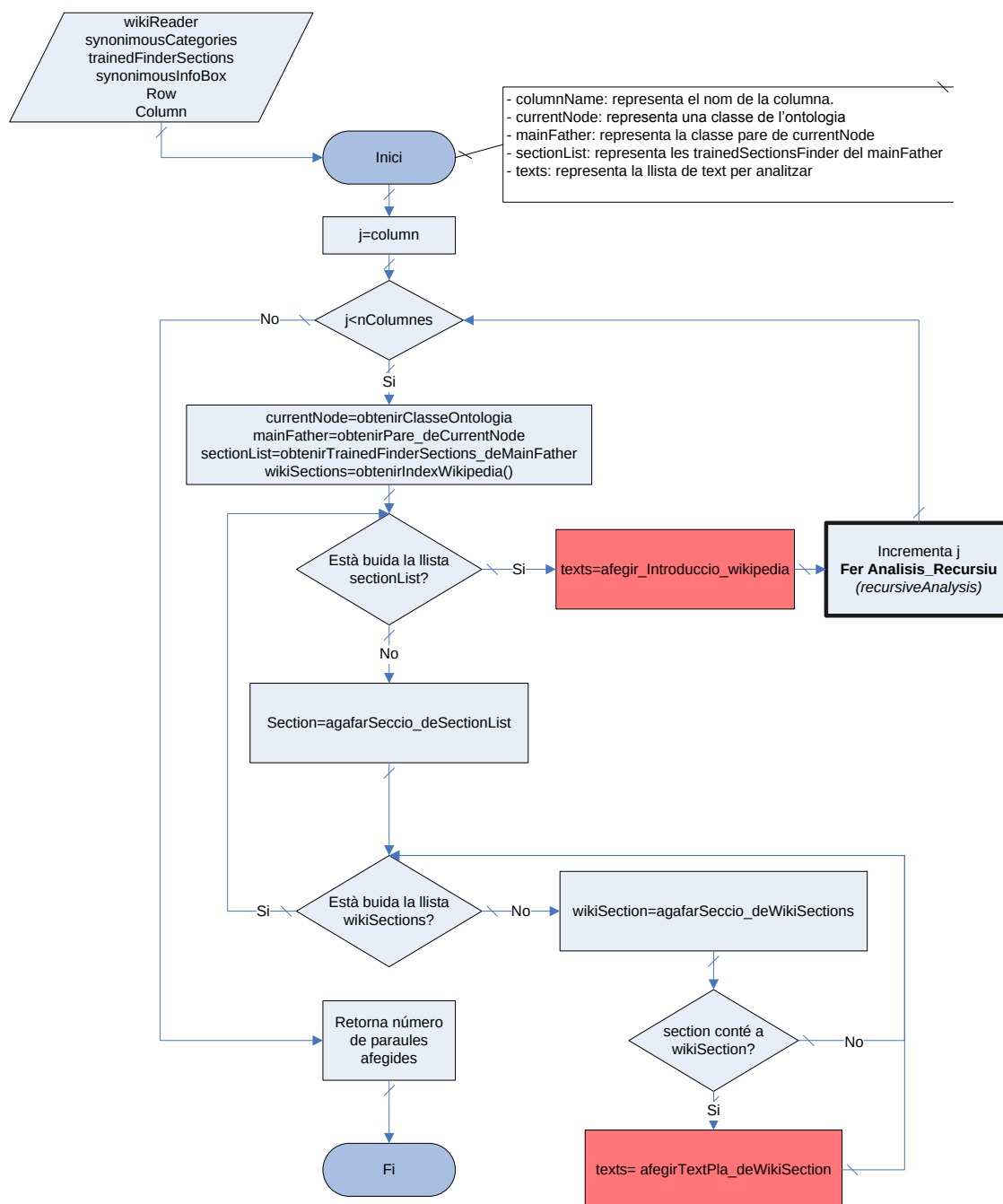


Figura 46 Diagrama de flux mètode *analyzeNode* d'*AnalizatorText*

La següent de les figures (Figura 47) mostra el diagrama de flux del mètode *recursiveAnalysis*, el comportament del qual és el mateix que en l'*AnalizatorCategories*. La Figura 48, que mostra el mètode *analyzeNode*, es diferencia de l'analitzador de categories en que quan troba una secció per examinar, afegeix el contingut d'aquesta secció a la llista de texts. Finalment, els mètodes *analyzeWithProperties* i *recursiveAnalysisProperties* són idèntics en la resta d'analitzadors (figures 49 i 50).

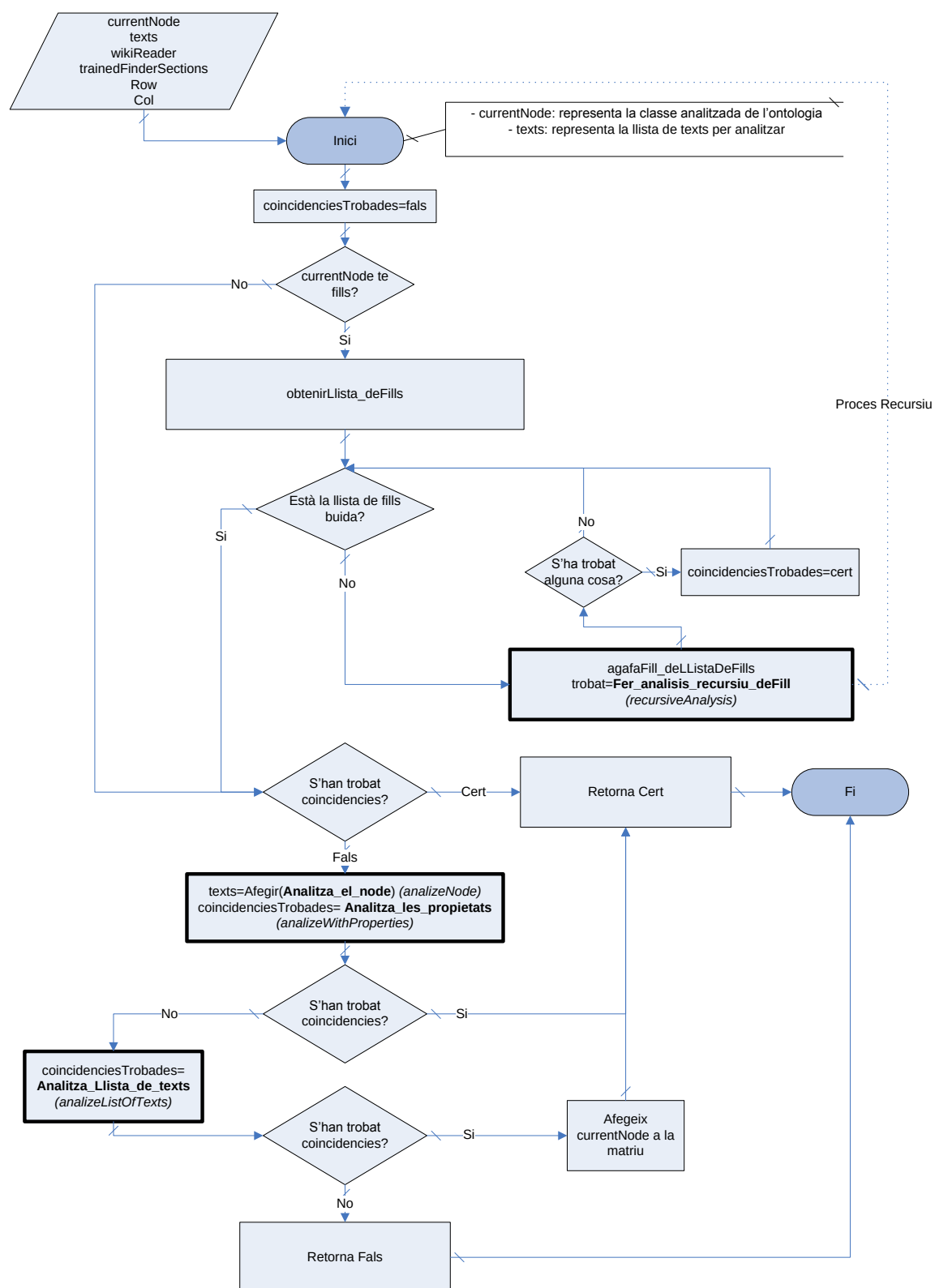


Figura 47 Diagrama de flux mètode *recursiveAnalysis* d'*AnalizatorText*

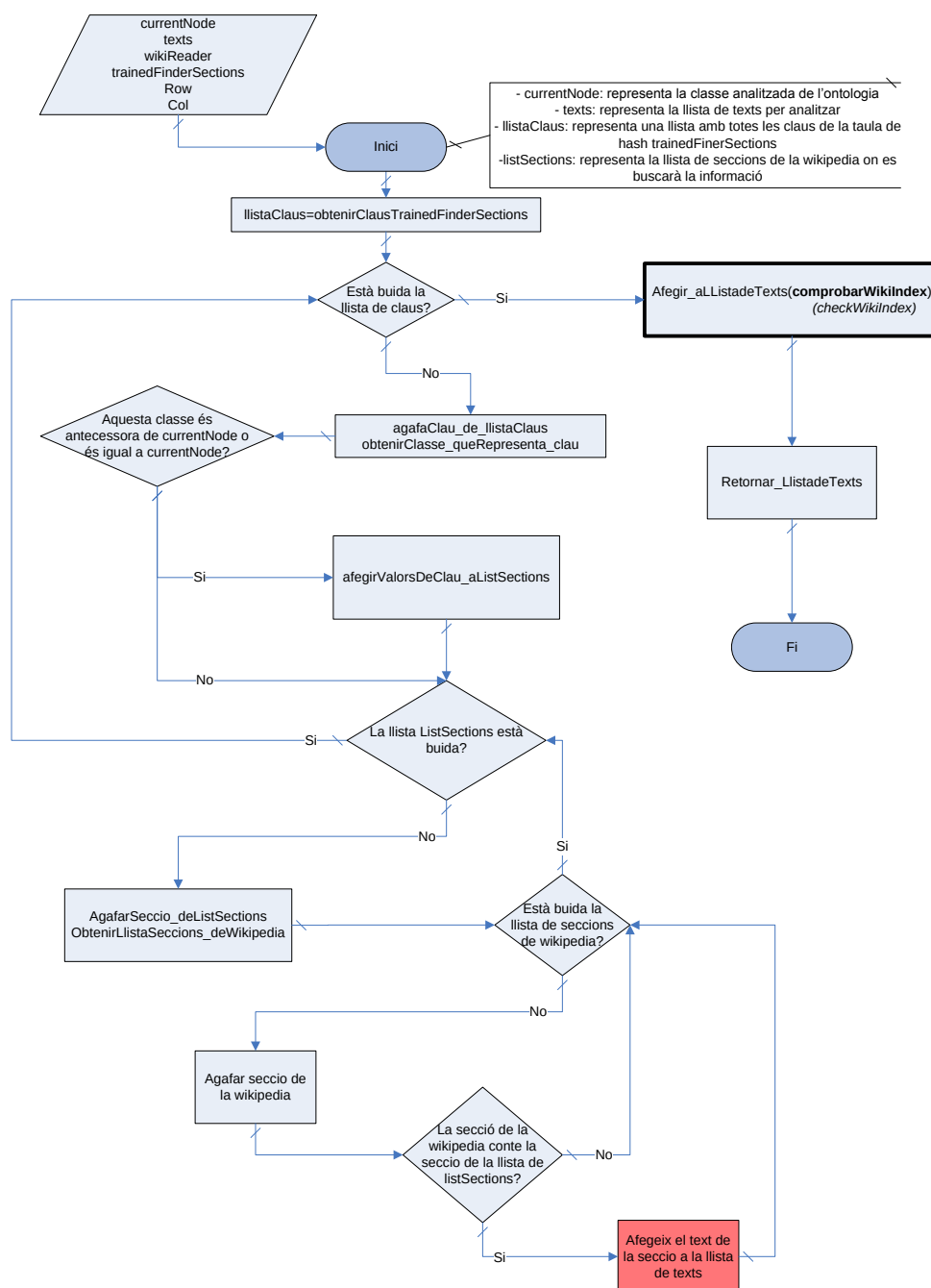


Figura 48 Diagrama de flux mètode *analyzeNode* d'*AnalizatorText*

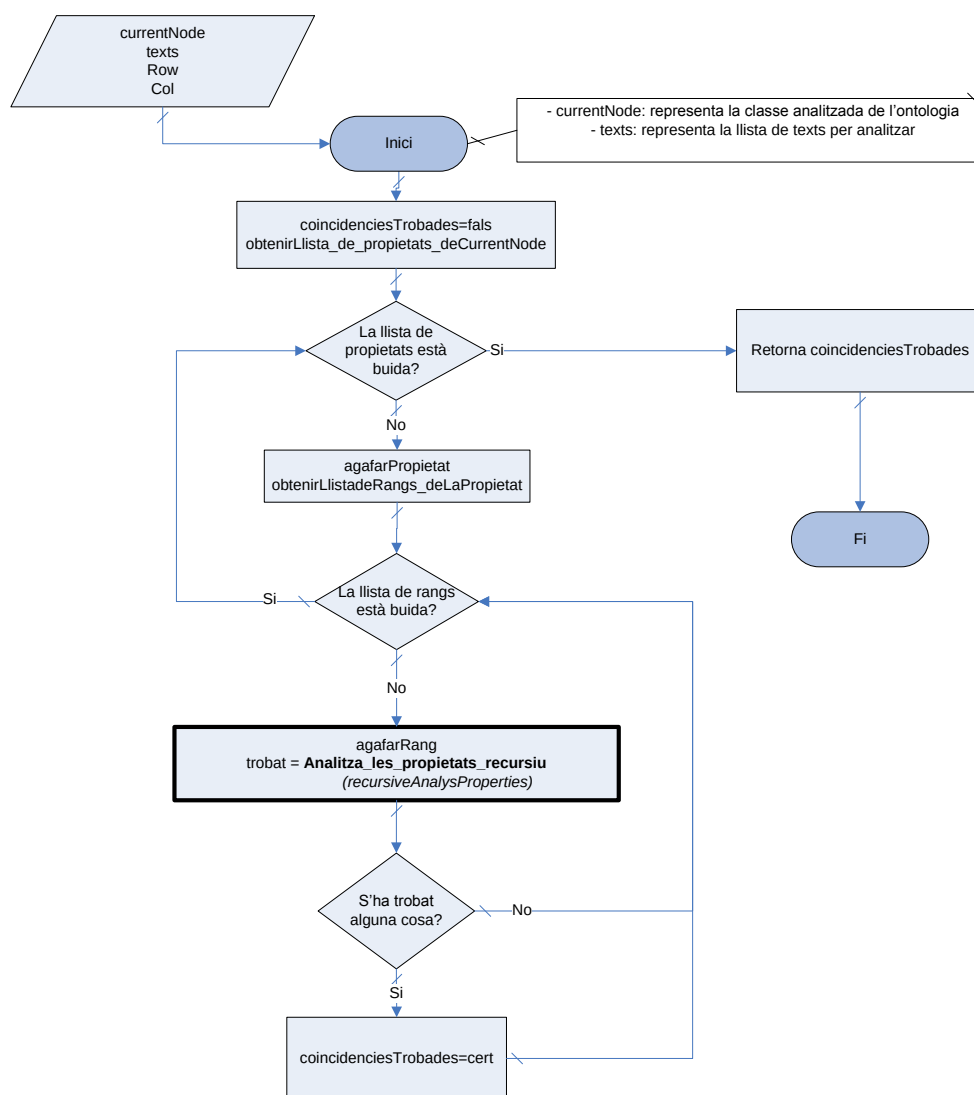


Figura 49 Diagrama de flux mètode *analyzeWithProperties* d'*AnalizatorText*

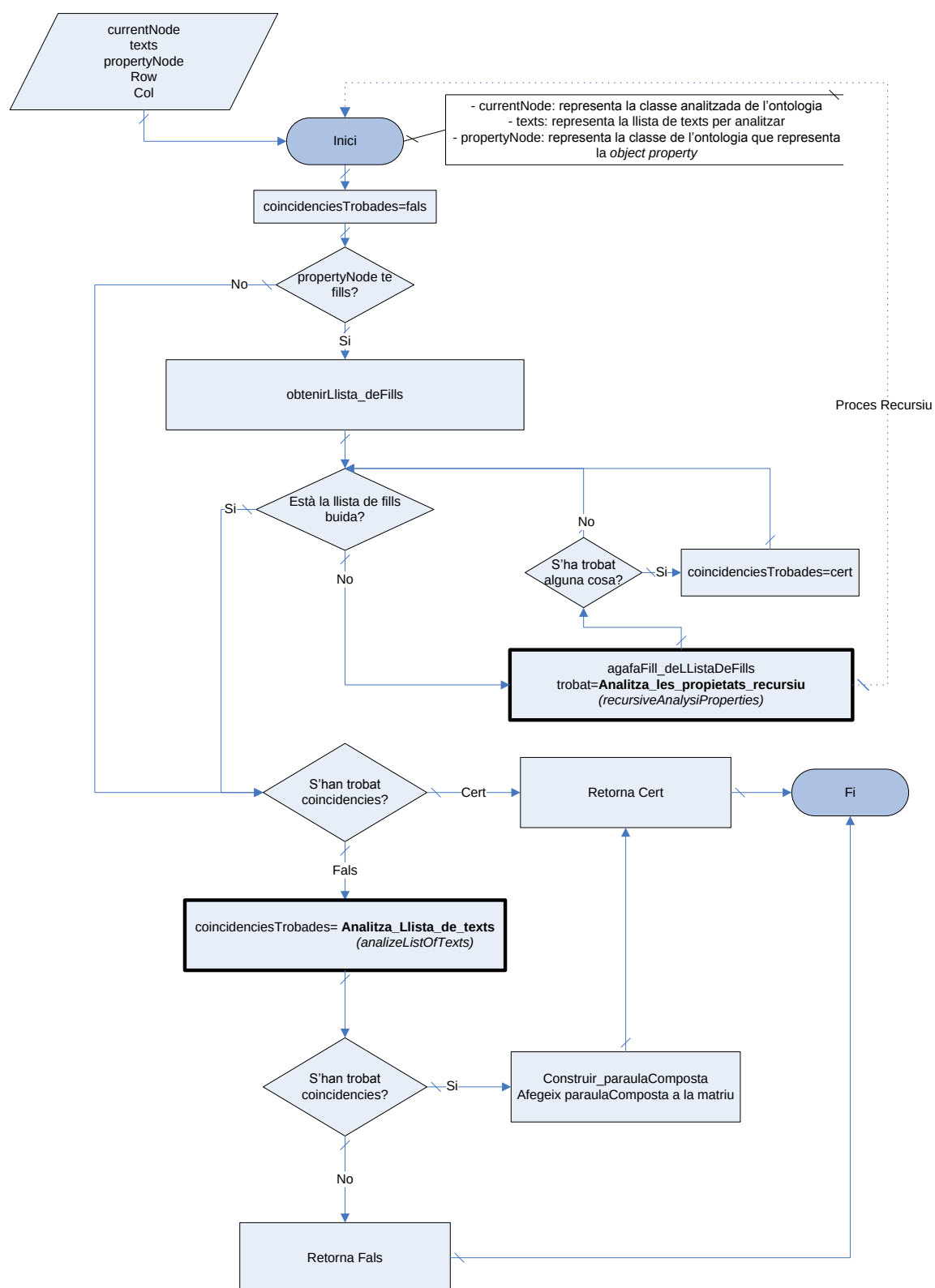


Figura 50 Diagrama de flux mètode *recursiveAnalysisProperties* d'*AnalizatorText*

4.2.2.2 Analitzador de categories (*AnalizatorCategories*)

Aquest analitzador és l'encarregat d'estudiar el contingut de la Wikipedia basant-se en les categories a les quals pertanyen les paraules de l'article analitzat que estan vinculades amb altres.

Seguint l'exemple de l'apartat anterior, es veurà com aquest analitzador aconseguirà les dades buscades. Recordem que en l'exemple esmentat es volia trobar informació sobre edificis culturals (*Cultural_Building*) i que només s'havia trobat la paraula *amphitheatre* situada al nivell dos de l'ontologia començant des de les fulles. Aquest analitzador començarà la cerca sense tenir en compte que anteriorment s'ha executat l'analitzador de text. Per tant, seguint els mateixos algorismes, es comportarà de la mateixa manera. En un moment donat, buscarà la paraula *Roman_Amphitheatre* (recordem que al text no apareixia aquest concepte). Com es pot veure en la Figura 51, la paraula trobada amb l'altre analitzador (*amphitheatre*) conté un link a un article de la Wikipedia on, si s'observen les categories a les que pertany aquest nou article, es veurà que són *Amphitheaters* i *Roman amphitheatre building*. En aquest moment, l'analitzador de categories decidirà afegir aquest nou concepte a la matriu de dades substituint al concepte anterior, ja que aquest és pare de l'actual i les dades s'afegeixen de nivell específic a nivell general.

Seguidament, igual que en l'analitzador anterior, es mostren els diagrames de flux dels mètodes principals. De tots aquests tan sols es diferencien de l'analitzador anterior els mètodes *extract* i *analyzeNode*. El primer d'ells es diferencia de l'anterior en el fet que en la llista de texts per analitzar s'afegeixen les categories a les que pertany una secció de la Wikipedia (veure Figura 52), en comptes del contingut de la secció (les categories s'obtenen a partir de les categories de tots els links que hi són a la secció). A més, en aquest mètode no s'analitza la introducció de la Wikipèdia, ja que es parla de conceptes molt generals que pertanyen a categories molt diverses, fet que pot induir a errors al resultat final. El segon dels mètodes divergeix de l'anterior de la mateixa manera, afegint a la llista de texts una llista de categories de la secció analitzada (veure Figura 53).

Finalment, els mètodes *recursiveAnalysis*, *analyzeWithProperties* i *recursiveAnalysisProperties* són completament iguals que els de l'analitzador de text (veure figures Figura 47, Figura 49 i Figura 50 respectivament).

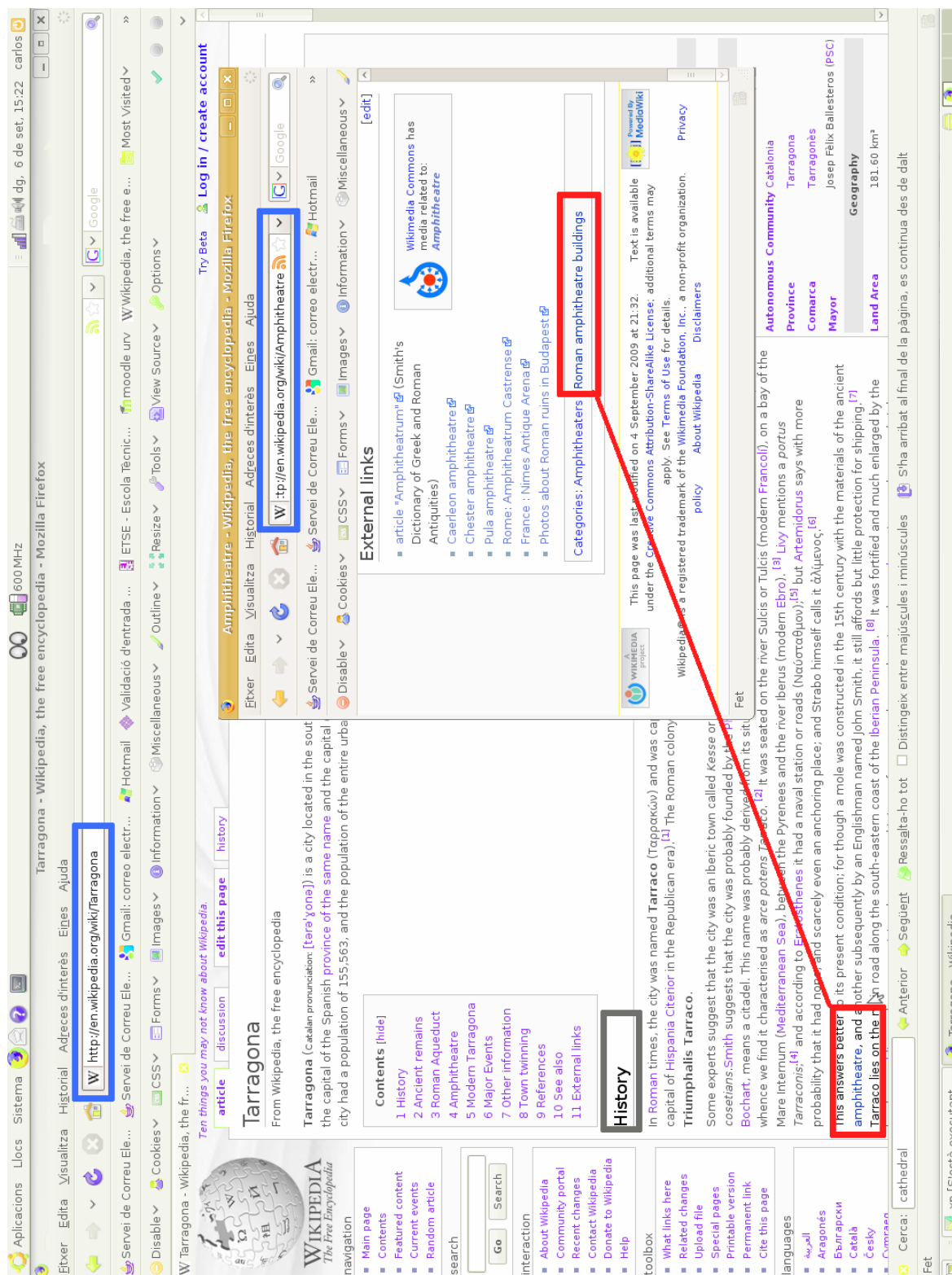


Figura 51 Exemple del procés d'extracció d'AnalitzatorCategories

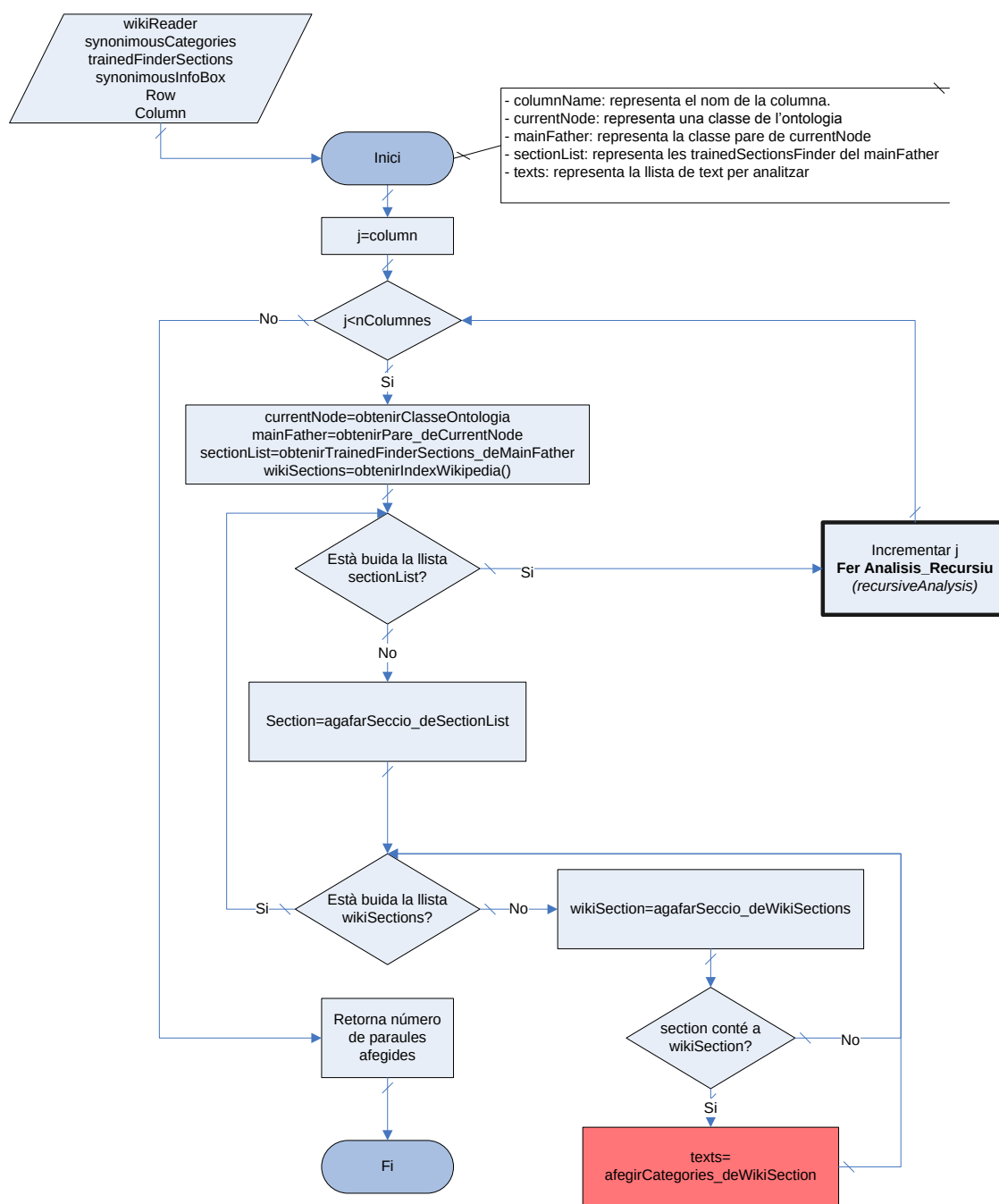


Figura 52 Diagrama de flux del mètode *extract* d'*AnalizatorCategories*

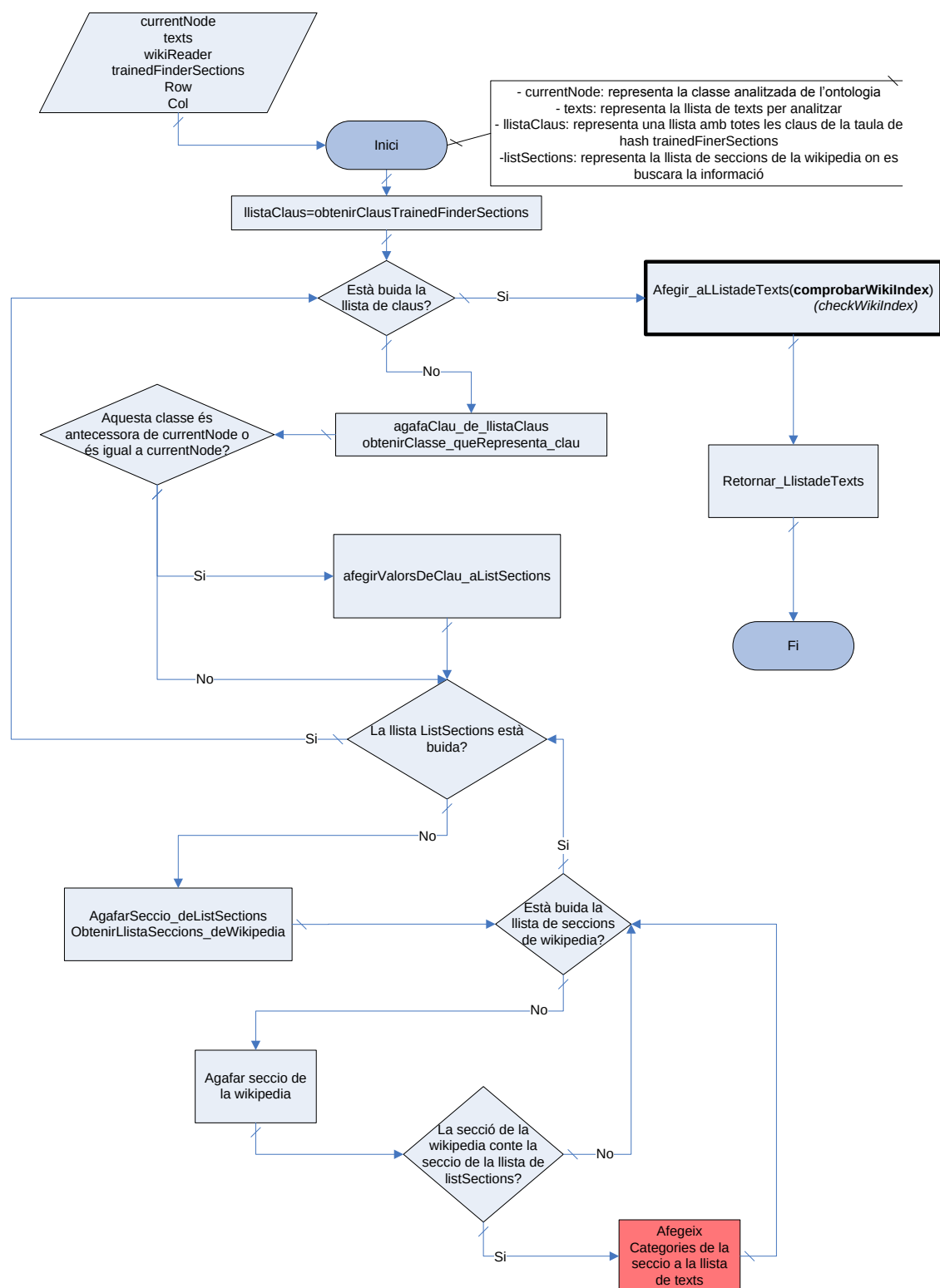


Figura 53 Diagrama de flux del mètode *analyzeNode* d'*AnalizatorCategories*

4.2.2.3 Analitzador d'imatges (*AnalizatorImages*)

Aquest analitzador només pretén complementar els anteriors ja que no està pensat per trobar moltes coincidències. Això es degut al fet que el sistema realitza la cerca basant-se només en els noms de les imatges que es troben a l'article. En la Figura 54 es pot veure el resultat generat per la Wikipedia en resposta a una petició que demana el nom de totes les imatges de l'article de Tarragona. Es mostra que els noms de les imatges estan separats pels símbols '.', '_' o '-' que es substitueixen pel caràcter 'espai' abans d'examinar els noms de les imatges. El principal problema d'aquest analitzador és que normalment els noms de les imatges no donen molta informació sobre l'objecte al qual representen, ja sigui perquè es troben en l'idioma natal de la ciutat de la que es parla en l'article consultat o perquè tenen noms propis que representen al objecte.

Continuant amb el mateix exemple que en els casos anteriors i suposant que encara no s'ha executat cap dels analitzadors, es podria veure com l'analitzador d'imatges obtindria la llista dels noms de les imatges de l'article de Tarragona, els filtraria i, en el moment de tractar-los, descobriria que existeix la paraula *amphitheatre* i, per tant, l'afegiria a la matriu de dades (veure Figura 55).

```

<api>
  <parse>
    <images>
      <img>E5327-Tarragona-Roman-wall.jpg</img>
      <img>Tarragona_Amphitheatre.jpg</img>
      <img>Spain-Tarragona-Portal-Sant-Antoni.00.A.jpeg</img>
      <img>Tarragona.Via.Imperial.jpg</img>
      <img>Spain.Tarragona.Catedral.Pla.Seu.00.JPG</img>
      <img>Spain.Tarragona.Catedral.Conques.02.JPG</img>
      <img>Tarragona.Pont_del_diable_aqüeducte.jpg</img>
      <img>TarragonaTorreScipioni.jpg</img>
      <img>Festa_major.jpg</img>
      <img>Tarragona.Pretorio.jpg</img>
      <img>Tarragona_Kathedrale.JPG</img>
      <img>E5320-Vista-de-Tarragona.jpg</img>
      <img>Bandera_de_Tarragona.svg</img>
      <img>Escut_de_Tarragona.svg</img>
      <img>Tarragona.Circo.Romano.jpg</img>
      <img>Flag_of_Spain.svg</img>
      <img>Tarragona9.jpg</img>
      <img>E5371-Tarragona-Aqueduct.jpg</img>
      <img>TarragonaCathedral2008.jpg</img>
      <img>Ambox_style.png</img>
      <img>Vista_desde_forti_de_la_reina.jpg</img>
      <img>Tarragona.noroeste.jpg</img>
      <img>Rambla2.jpg</img>
      <img>Flag_of_France.svg</img>
      <img>Flag_of_Italy.svg</img>
      <img>Flag_of_UK.svg</img>
      <img>Flag_of_Austria.svg</img>
      <img>Flag_of_Russia.svg</img>
      <img>Commons-logo.svg</img>
      <img>Flag_of_Catalonia.svg</img>
    </images>
  </parse>
</api>

```

Figura 54 Resultat generat per una petició a la Wikipedia que demana els noms de les imatges de l'article de Tarragona

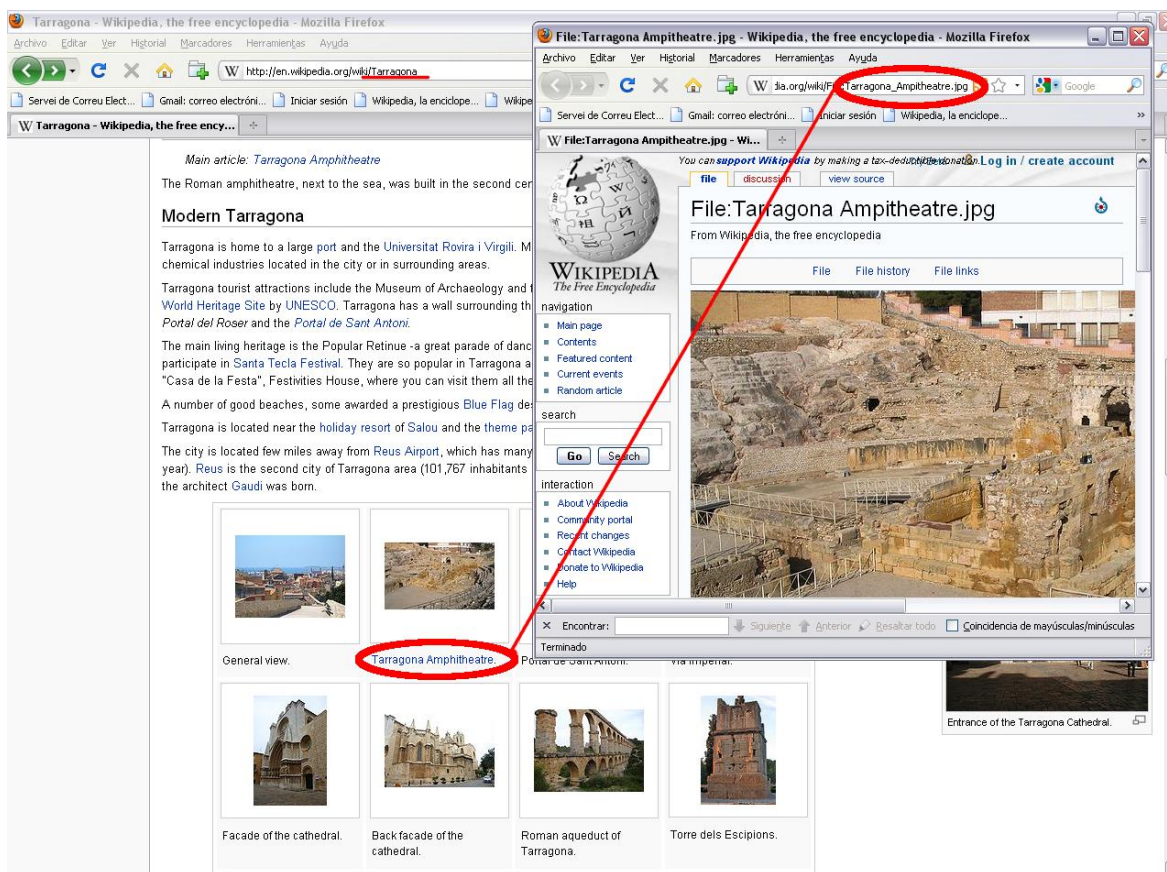


Figura 55 Exemple del procés d'extracció d'AnalitzadorImages

Respecte als mètodes implementats en aquest analitzador, només se'n diferencien els dos primers (*extract* i *recursiveAnalysis*), el tercer mètode (*analyzeNode*) no existeix i la resta són totalment iguals que en els casos anteriors.

El primer d'ells es diferencia en el fet que només recorre la matriu per columnes i per cadascuna afegeix a la llista de textos per analitzar els noms de totes les imatges, prèviament filtrades, que es troben en l'article consultat i, posteriorment, s'inicia el procés d'anàlisi recursiu (*recursiveAnalysis*) (veure Figura 56).

En el segon cas la diferència és que no es crida al mètode *analyzeNode* perquè no és necessari saber quins textos analitzar ja que només s'analitzen els noms de les imatges (Figura 57).

Els mètodes *analyzeWithProperties* i *recursiveAnalysisProperties* es poden veure en les figures Figura 49 i Figura 50.

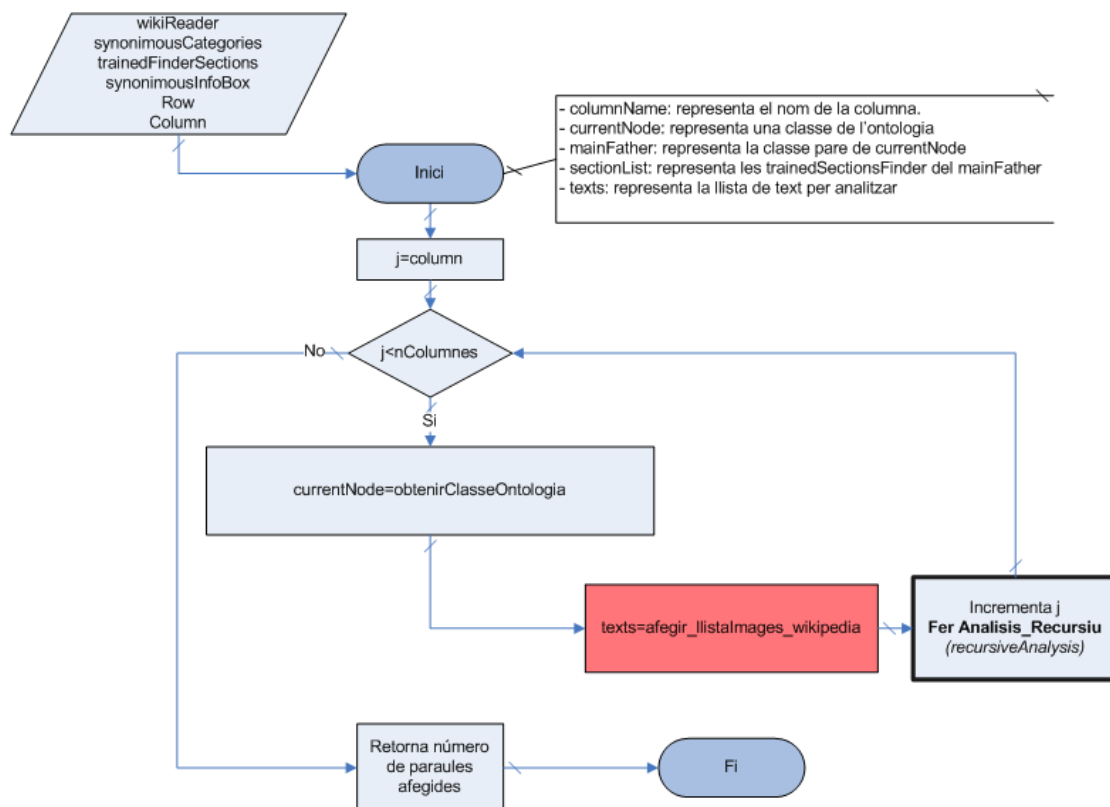


Figura 56 Diagrama de flux del mètode *extract* d'*AnalizadorImages*

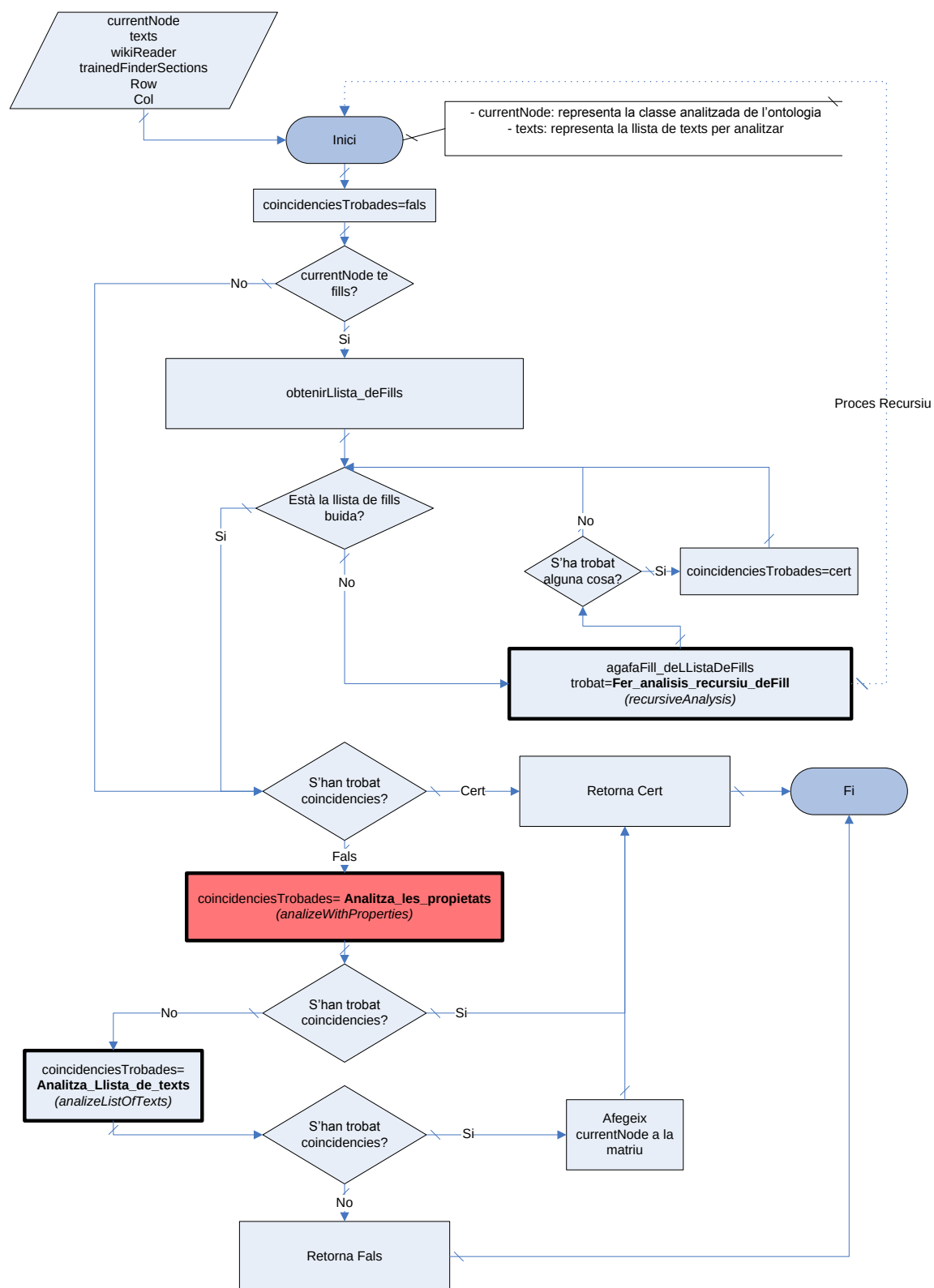


Figura 57 Diagrama de flux del mètode *recursiveAnalysis* d'*AnalizatorImages*

4.2.2.4 Analitzador de taules (*AnalizadorInfoBox*)

L'últim dels analitzadors es comporta de manera diferent a tots els anteriors. El seu objectiu es extreure dades a partir de les *infoBox* de la Wikipedia (veure Figura 28 i secció 3.4.1). El principal problema que presenta l'extracció de dades a partir d'aquesta font és el fet que aquestes taules es basen en plantilles de la Wikipedia i n'hi ha de molt diverses. A més a més, cada plantilla guarda els atributs i valors de les dades amb noms diferents, la qual cosa dificulta encara més la cerca i filtrat d'informació. També s'ha de tenir en compte que les dades emmagatzemades en les taules es guarden en formats molt diferents segons la plantilla que utilitzen. Per exemple, la superfície d'una ciutat pot estar guardada tant en quilòmetres quadrats com en milles al quadrat.

Existeixen dos algorismes principals en el procés d'extracció. El primer és el mètode *extract* i en ell es crea un diccionari a partir de la *infoBox* (Figura 58). Aquest diccionari està format per una paraula clau, que es correspon a l'atribut que es troba a la taula (paraula situada al costat esquerre del símbol '=' de la Figura 58) i per un o més valors (costat dret del símbol '='). Un cop s'ha creat el diccionari, es procedeix a fer l'anàlisi (mètode *analyzeInfoBox*). En la Figura 59 es pot veure el diagrama de flux d'aquest procediment. El segon mètode és *analyzeInfoBox* i és el responsable de fer l'anàlisi de les dades i afegir els resultats trobats a la matriu. El sistema comença recorrent les columnes de la matriu per conèixer les dades que s'han de cercar i, mitjançant l'ajuda del diccionari creat pel mètode anterior, es decideix on buscar les dades. Un cop es troba algun resultat, el primer que es fa és filtrar-lo, ja que pot contenir comentaris. A continuació, i en cas que sigui necessari, es tracta aquest valor i es converteix a un format comú per a totes les dades. Finalment, s'afegeix a la matriu.



```

{{Spanish city
|image_skyline = E5320-Vista-de-Tarragona.jpg
|image_skyline_size = 250px
|image_skyline_caption = View of Tarragona
|native_name = Tarragona
|spanish_name = Tarragona
|nickname =
|city_motto =
|city_motto_means =
|image_flag = Bandera de Tarragona.svg
|image_flag_size = 150px
|image_coat_of_arms = Escut de Tarragona.svg
|image_coat_of_arms_size = 80px
|image_city_map =
|image_city_map_size = 280px
|image_city_map_caption = Location of Gijón
|lat_long = {{coord|41|06|56.51|N|1|14|58.54|E|}}
|time_zone = [[CET]] (GMT +1)
|time_zone_summer = [[CEST]] (GMT +2)
|founded = [[5th century BC]]
|native_language = Catalan
|community = Catalonia
|community_link = Catalonia
|province = Tarragona
|province_link = Tarragona (province)
|comarca = Tarragonès
|comarca_link = Tarragonès
|divisions =
|neighborhoods =
|mayor = Josep Fèlix Ballesteros
|political_party = PSC
|political_party_link = Spanish Socialist Workers' Party
|area = 181.60
|altitude = 68
|population = 155,563
|date-population = 2009
|population-ranking =
|density = 1,511.40
|date-density = 2009
|website = http://www.tarragona.cat/
|postal_code = 43001 - 43008
|area_code =
}}

```

Figura 58 Exemple d'*InfoBox* en format raw

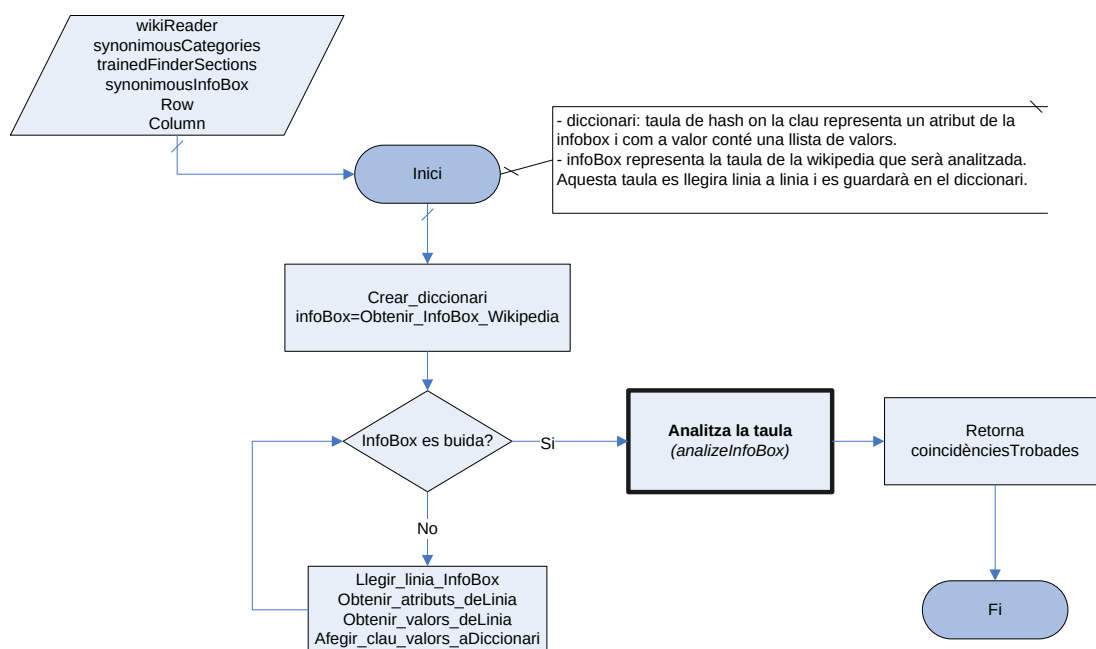


Figura 59 Diagrama de flux del mètode *extract* d'*AnalizatorInfoBox*

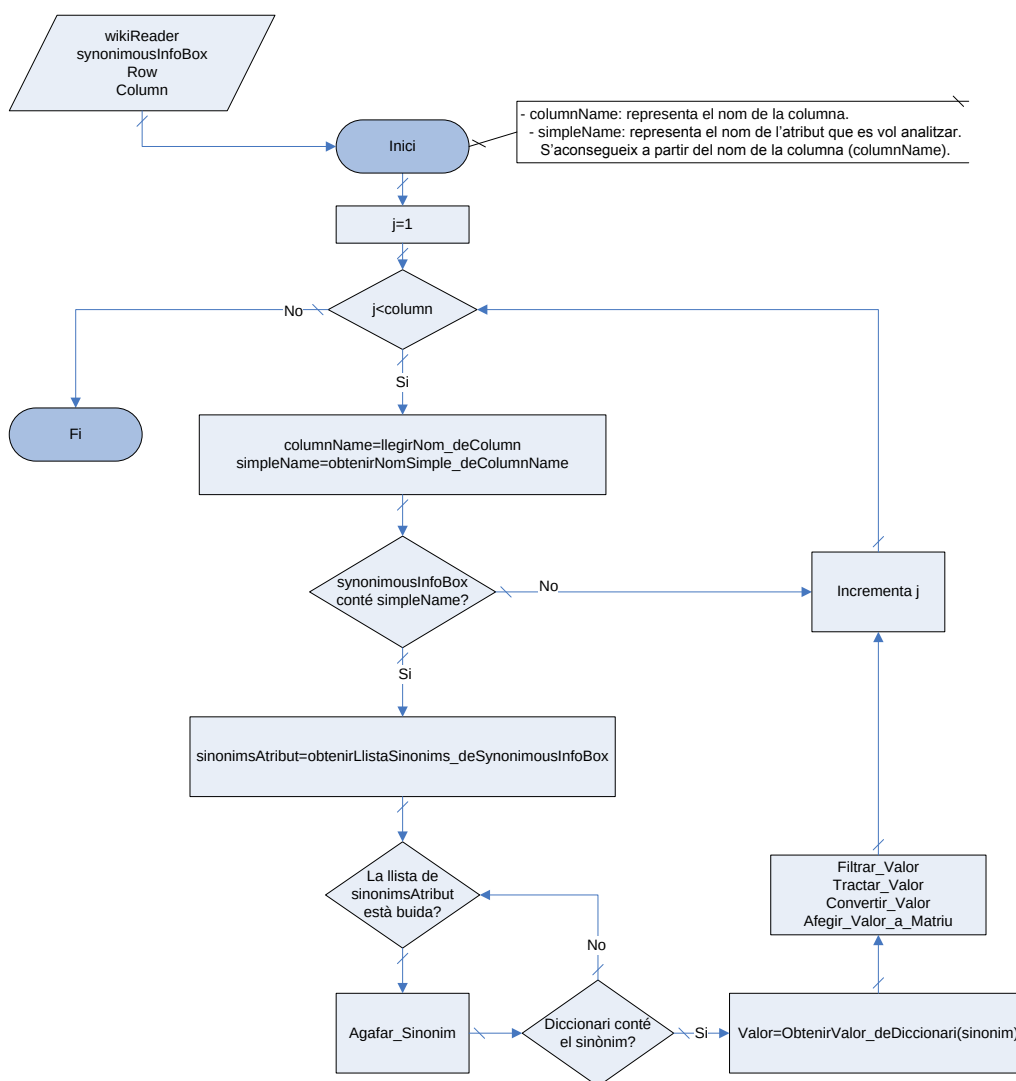


Figura 60 Diagrama de flux del mètode *analyzeInfoBox* d'*AnalizatorInfoBox*

4.2.3 Lectors (package reader)

Aquest paquet facilita tots els mitjans necessaris per llegir planes de la Wikipedia. Està format per les classes *IWikiReader*, *WikiReader*, *WikipediaTextReader* i *ExtractorReader*.

En la Figura 61 es pot veure com, igual que en el cas dels analitzadors, existeix una *interface* (*IWikiReader*) que defineix tots els mètodes que s'han d'implementar. La següent classe es tracta d'una classe abstracta (*WikiReader*) que utilitza aquesta *interface*. Finalment, *WikipediaTextReader* implementa tots els mètodes necessaris per llegir la Wikipedia i guardar totes les seccions, taules, categories, etc.

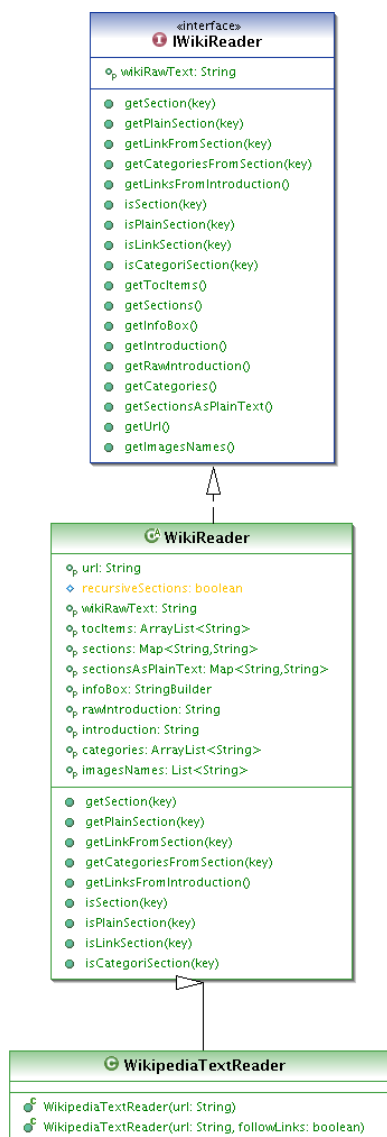


Figura 61 Herència de classes del paquet *reader*

En la següent figura (Figura 62) es mostra un diagrama on es pot veure com treballa aquest lector. El primer que fa és obtenir l'article en format *raw* (format d'edició de la Wikipedia) i a continuació el llegeix i el tracta: busca la taula *InfoBox* i l'emmagatzema, busca la introducció de la Wikipedia i la guarda amb format de text pla i text *raw*, aconsegueix l'índex de seccions de l'article i, per a cadascuna, desa el seu contingut amb els mateixos formats que en el cas anterior. Si no existeix cap índex de seccions, la plana de la Wikipedia no s'analitza. Afegeix les categories a les que pertany l'article de la Wikipedia i els noms de les imatges que hi ha i, finalment, si està activada la variable

followLinks, recorre totes les seccions: per a cadascuna es crea una llista amb els links interns i per cada link s'obtenen totes les categories a les quals pertany l'article que representa.

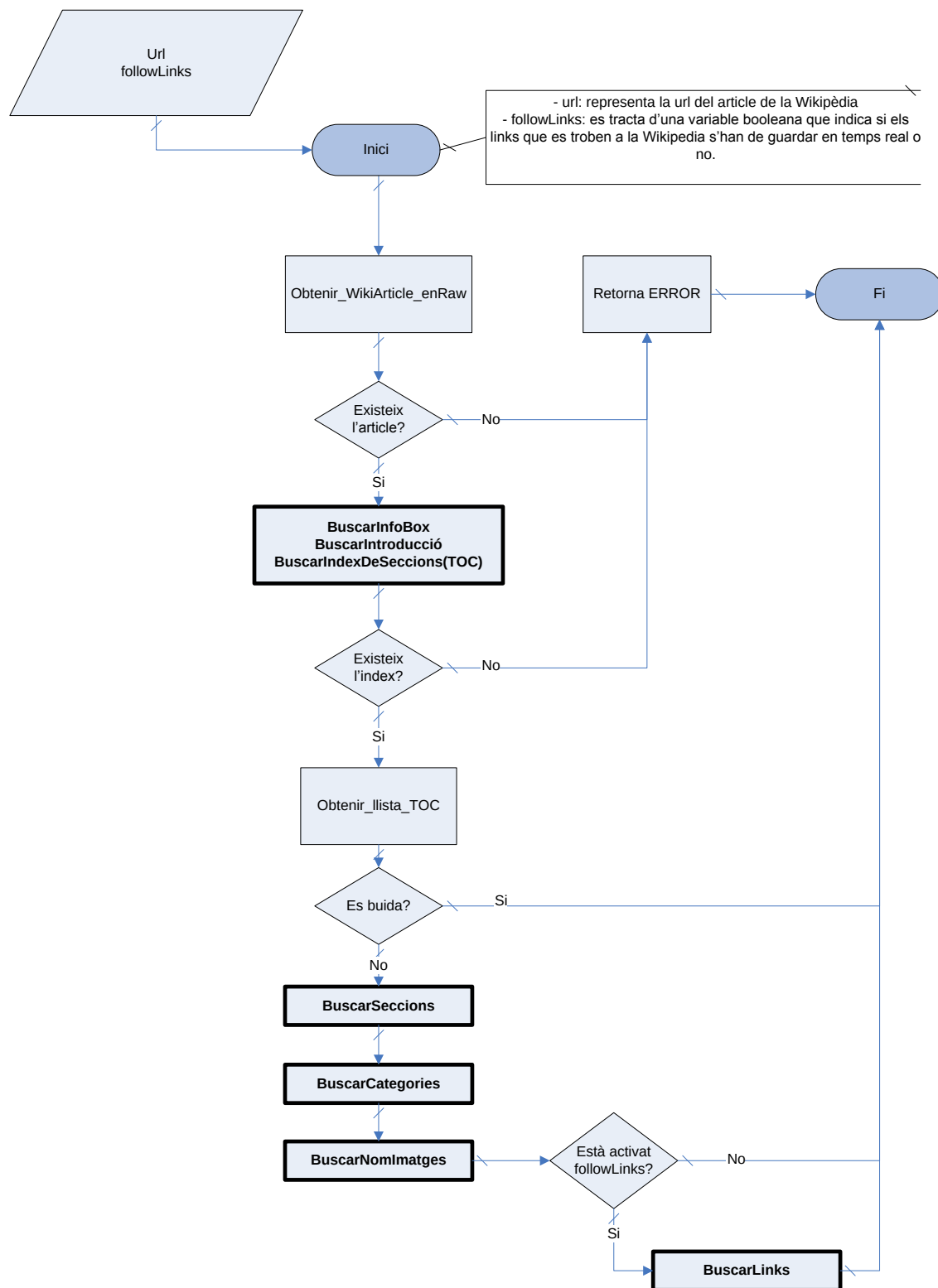


Figura 62 Diagrama de flux de *WikipediaTextReader*

En la Figura 61 es poden veure tots els mètodes que proporciona qualsevol objecte del tipus *reader*. Els principals són:

- *getSection(key)*: retorna la secció que identifica la clau en forma de text *raw*.
- *getPlainSection(key)*: retorna la secció que identifica la clau en forma de text pla.
- *getLinksFromSections(key)*: retorna la llista de links interns que es troben a la secció identificada per la clau.
- *getCategoriesFromSection(key)*: retorna una llista amb totes les categories a les quals pertanyen els links interns que es troben a la secció identificada per la clau.
- *getTocItems()*: retorna una llista amb l'índex de seccions de la Wikipedia.
- *getInfoBox()*: retorna la *infoBox* de la Wikipedia.
- *getIntroduction()*: retorna la introducció de l'article en forma de text pla.
- *getImageNames()*: retorna una llista amb tots els noms de les imatges que es troben a la Wikipedia.

La classe *ExtractorReader* (Figura 63) és l'encarregada d'agrupar tots els elements que intervenen durant el procés d'anàlisi i extracció i proporciona una interfície (mètode *extract*) per iniciar aquesta última. A més a més, facilita informació estadística sobre el total de paraules trobades per cada analitzador, les que ha afegit cadascun d'ells i el temps que han trigat durant l'anàlisi.

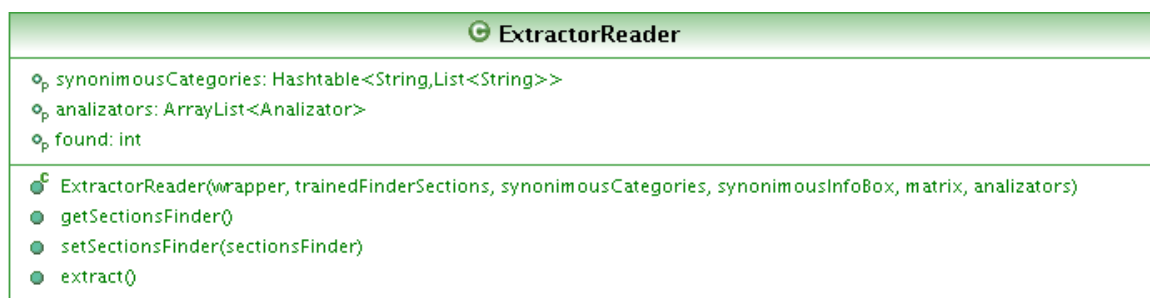


Figura 63 Mètodes de la classe *ExtractorReader*

4.2.4 Programa principal (package Main)

El paquet *Main* només conté dos classes principals (*Main1* i *Main2*) usades per executar dos jocs de proves diferents. En el primer s'analitzen 150 ciutats mitjançant els analitzadors de text, categories i imatges. El segon prova el funcionament de l'analitzador d'*InfoBox* per a les mateixes ciutats.

S'ha decidit afegir en aquest document una secció dedicada a aquest paquet per poder explicar com s'hauria de crear qualsevol aplicatiu que treballi amb aquest sistema.

Per una part, sota el directori d'execució ha d'existir un altre que contingui un fitxer de text amb el nom de totes les ciutats que s'hagin d'analitzar (*cities*), una matriu, generada amb l'aplicatiu de tractament de matriu implementat en aquest projecte i amb format *.ser*, amb les columnes inicialitzades segons les dades que es vulguin buscar durant l'anàlisi (*matrix.ser*) i un fitxer amb l'ontologia turística utilitzada per guiar la cerca (*TourismOWL.owl*). Com a resultat, tal com es mostra a la Figura 64, el sistema generarà dos fitxers en els quals es guardaran les dades extretes (*matrixSaved.ser*) i les estadístiques sobre l'anàlisi (*statistics.ser*).

Per l'altra part, la classe que implementi el programa principal ha de contenir un objecte del tipus *ExtractorReader*, un objecte del tipus *Wrapper* i les tres taules de *hash*

amb les dades de *trainedFinderSections*, *synonymousCategories* i *synonymousInfoBox* (veure Figura 65).

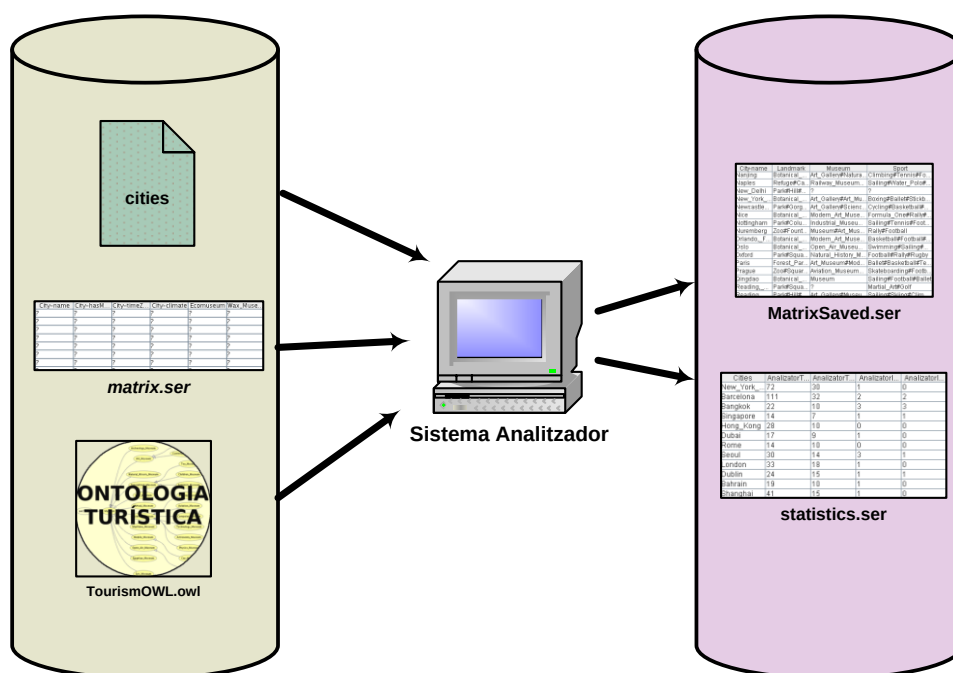


Figura 64 Funcionament programa principal



Figura 65 Parts necessàries del programa principal

4.2.5 Útils (package utils)

L'últim dels paquets que s'explica en aquest document és el paquet *utils*. Aquest conté quatre classes implementades per oferir una sèrie d'eines de suport, utilitzades pels diferents processos que conformen el sistema. Aquestes classes són: *Filters*, *Parser*, *Query* i *ProcessXML* (veure Figura 66).

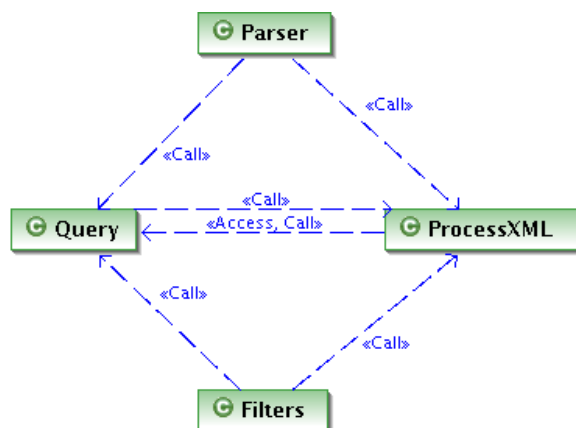


Figura 66 Classes del paquet *utils*

La classe *Filters* proporciona mètodes per filtrar *Strings*. En algunes ocasions és necessari eliminar els links interns d'un text de la Wikipedia, les dades numèriques, les referències externes, els espais en blanc o els comentaris que es troben dintre de l'article. Tots els mètodes d'aquesta classe (Figura 67) estan implementats amb l'ús de *javaregex*⁵.

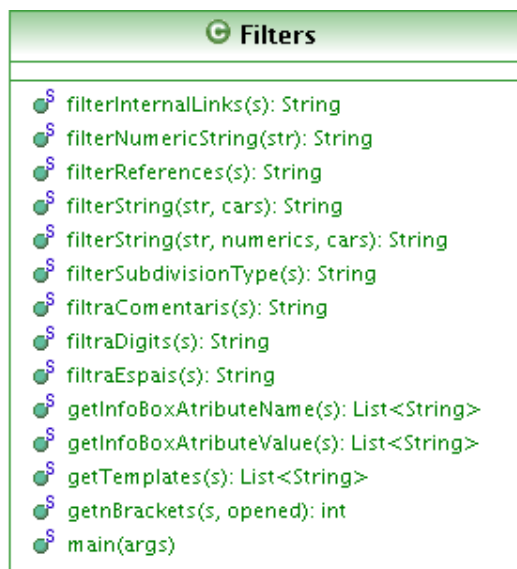


Figura 67 Mètodes de la classe *Filters*

La classe *Parser* s'utilitza per convertir dades d'un format a un altre. Per exemple, es pot desitjar convertir un text de la Wikipedia que es troba en format *raw* a text pla o, donada una latitud i una longitud, obtenir una ubicació geomètrica en forma de graus, minuts i segons. En la Figura 68 es poden veure alguns dels mètodes que ofereix aquesta classe.

⁵ *javaregex* és una API del Java que proporciona mètodes i tècniques avançats per manejar des d'expressions regulars senzilles fins a expressions molt complexes.

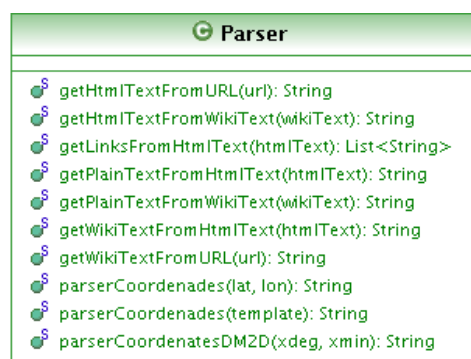


Figura 68 Mètodes de la classe *Parser*

La classe *Query* és l'encarregada de gestionar totes les peticions que es fan a la Wikipedia. En aquesta classe es poden distingir dos parts: les variables estàtiques i els mètodes estàtics. Un dels mètodes més importants d'aquesta classe és el mètode *getXmlPageParam* què, mitjançant el paràmetre *param* proporciona un nivell de flexibilitat molt alt per realitzar les consultes. Aquest paràmetre es correspon a alguna de les variables estàtiques que s'han comentat. El mètode rep la *URL* de la plana sobre la que es vol fer una consulta i un paràmetre que especifica quin tipus d'informació es desitja i, finalment, el mètode retorna el resultat en format *XML*. Algunes de les parts que es poden demanar són l'índex de seccions, el nom de les imatges, els links externs o el text. En la Figura 69 es veuen els mètodes i les variables d'aquesta classe.



Figura 69 Mètodes de la classe *Query*

L'última classe és la classe *ProcessXML* i, com el seu nom indica, s'utilitza per processar els resultats *XML* generats per la classe anterior (veure relacions entre aquestes dos classes en la Figura 66). Cada mètode rep per paràmetre un text en *XML* que pot contenir el resultat de peticions sobre categories, coordenades, noms d'imatges, seccions de la Wikipedia, etc. Com a resultat, cada mètode generarà unes dades diferents en funció del text que s'hagi processat: en el cas de les categories es retornarà una llista de categories, en el cas de les seccions es retornarà una llista amb els noms de totes les seccions, i així amb cadascun dels mètodes. En la Figura 70 es mostren tots aquests mètodes i els tipus de dades que retorna cadascun.

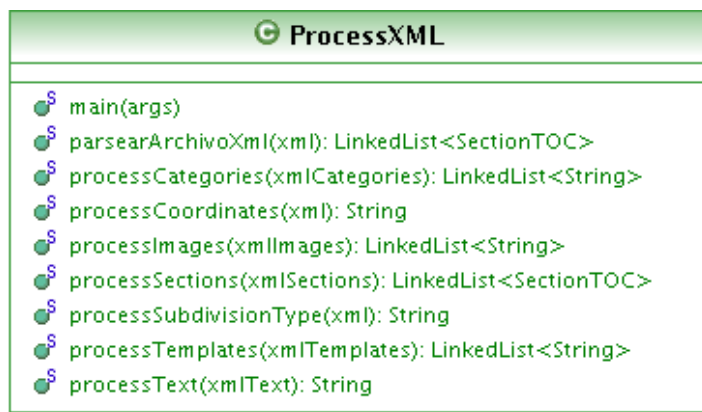


Figura 70 Mètodes de la classe *ProcessXML*

4.3 Limitacions generals

En aquesta última secció, es vol donar una visió general de les limitacions tant de l'eina de tractament de matriu com del sistema analitzador.

Respecte l'**Eina de tractament de matriu** es coneixen les següents limitacions:

- No soporta l'hèrència multiple.

També es poden espentar els següents problemes:

- Si el domini d'una *object property* pertany a més d'una classe de l'ontologia, l'eina de tractament de matriu situa aquesta propietat com a propietat de la classe *owl:thing*. Aquest problema deriva de la classe *Wrapper*.
- Amb les *subproperties* succeeix el mateix.
- Suposem que, mitjançant l'eina, es selecciona una classe de l'ontologia i alguna de les seves propietats; per tant, s'afegirà en una columna de la matriu el nom de la classe seleccionada i, en la columna següent, l'atribut marcat. Suposem ara que es vol agregar a la matriu la classe filla de la columna afegida anteriorment; el funcionament de l'aplicatiu eliminarà tant la columna pare de la matriu com la de l'atribut seleccionat i la substituirà per la classe filla. El problema es troba en el fet que la classe filla, segons la definició d'ontologia, ha d'heretar els atributs del pare i, si els del pare estan activats, també ho haurien d'estar en els fills: aquest problema també deriva de la classe *Wrapper*.

Respecte el **Sistema analitzador** les limitacions conegudes són:

- Les classes de l'ontologia formades per més d'una paraula no poden estar separades per espais. S'ha d'utilitzar el caràcter '_'.
- Actualment el sistema només utilitza la comparació de *Strings* per cercar la informació.
- La informació trobada pot ser errònia perquè:
 - No es fa anàlisi semàntic.
 - La Wikipedia està editada per usuaris de tot el món i pot contenir errors.
- Degut a la gran quantitat d'inter-relacions entre els articles de la Wikipedia el cost temporal d'anàlisi és molt elevat.

Al següent capítol es mostraran els resultats obtinguts pel sistema, s'analitzarà la qualitat en un subconjunt seleccionat de ciutats i s'identificaran diversos motius que poden portar a tenir resultats incomplets i/o incorrectes.

5 RESULTATS I JOC DE PROVES

Per tal de determinar la qualitat dels resultats finals, en aquesta secció es farà una anàlisi manual basada en l'aspecte quantitatiu. L'objectiu és proporcionar mesures comparables de qualitat; en aquest cas, s'utilitzaran les típiques en *IR* (Recuperació d'informació), que són el *recall*, la *precision* i *F-measure*.

- *Recall*: Mostra, de tot el coneixement que es podia trobar, quina quantitat s'ha extret.

$$\text{Recall} = \frac{\#Conceptes\ Bons\ Trobats}{\#Bons\ Trobats + \#Bons\ No\ trobats}$$

- *Precision*: Especifica fins a quin punt el coneixement que s'ha extret és correcte.

$$\text{Precision} = \frac{\#Conceptes\ Bons\ Trobats}{\#Bons\ Trobats + \#Dolents\ Trobats}$$

- *F-measure*: Normalment la *precision* i el *recall* són inversament proporcionals, ja que en augmentar la quantitat de dades trobades és més possible generar errors. *F-measure* calcula una mitja ponderada entre els conceptes anteriors per facilitar la comparació entre ells. S'utilitza per donar una visió general sobre el rendiment global del sistema recuperador de dades.

$$F\ Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

En l'apartat 5.1, es mostraran els resultats obtinguts i es farà una breu anàlisi sobre la fiabilitat i rendiment del sistema analitzador. A continuació s'interpretaran els resultats i es realitzarà una valoració general.

5.1 Resultats

Per poder valorar el sistema s'han escollit quatre ciutats i s'han analitzat manualment. Aquestes són *Barcelona*, *Bath (Somerset)*, *Cardiff* i *Marseille*.

En la següent taula (Figura 71), es mostren els valors obtinguts per a les quatre ciutats esmentades. A continuació, en la llegenda, es pot veure el tipus d'informació segons els colors:

- Roig: *Conceptes Dolents Trobats*.
- Negre: *Conceptes Bons Trobats*.
- Verd: *Conceptes Bons No Trobats*.
- Blau: Es consideren com a *Conceptes Bons Trobats* encara que el sistema hagués pogut detectar un concepte de l'ontologia de nivell inferior. Per exemple, *Park* en comptes de *Urban_Park*. Això significa que els resultats són bons però podrien ser millors.

City-name	Landmark	Museum	Sport	Building
Cardiff	- Beach - Canal - Hill - Lake - Megalithic - Mountain - Nature_Reserve - Park - Pyramid - River - Memorial_LandMark	- Art_Museum - Museum - Natural_History_Museum - Open_Air_Museum	- Badminton - Ballet - Basketball - Boxing - Climbing - Cricket - Football - Hockey - Ice_Hockey - Paddle - Rally - Rugby - Sailing - Skiing - Snowboarding - Squash - Surfing - Swimming - Table_Tennis - Tennis - Judo - Kaiaiking	- Abbey - Cathedral - Chapel - Church - Fair - Fort - Hotel - House - Library - Market - Mosque - Opera - Palace - Pool - School - Shop - Shopping_Area - Shopping_Centre - Shopping_Street - Skyscraper - Stadium - Synagogue - Temple - Theatre - Tower - University - Music_School
Bath,_Somerset	- Beach - Botanical_Garden - Bridge - Column - Hill - Ionic_Column - Park - River - Square - Statue - Stone_Bridge - Terrace	- Art_Gallery - Art_Museum - Astronomy_Museum - Biographical_Museum - Science_Museum	- Badminton - Basketball - Bowling - Cricket - Cycling - Football - Golf - Hockey - Judo - Rugby - Skateboarding - Swimming - Tennis - Volleyball	- Abbey - Amphitheatre - Cathedral - Church - Fair - Forum - Golf_Course - Hotel - House - Library - Luxury_Hotel - Market - Monastery - Opera - Palace - Parish - Royal_Residence - School - Shop - Shopping_Centre - Shopping_Street - Stadium - Temple - Theatre - Tower - University - Supermarket - Popular_Running
Marseille	- Beach - Botanical_Garden - Canal - Cave - Column - Crypt - Fountain - Hill - Mountain - Obelisk - Square - Statue - Terrace - Triumphal arch	- Art_Gallery - Art_Museum - Contemporary_Art_Museum - Maritime_Museum - Modern_Art_Museum - Natural_History_Museum - Archeology_Museum	- Ballet - Cycling - Football - Golf - Rugby - Sailing - Swimming - Windsurfing	- Abbey - Baroque_Chapel - Basilica - Byzantine_Basilica - Church - Fair - Fort - Golf_Course - Hotel - House - Library - Market - Medieval_Cathedral - Monastery - Opera - Parish - Pool - Prison - Shopping_Area - Stadium - Store - Temple - Theatre - Tower - Velodrome - Mall - Fashion Shop - Byzantine Cathedral - Palace

City-name	Landmark	Museum	Sport	Building
Barcelona	- Beach	- Archeology_Museum	- Basketball	- Abbey
	- Crypt	- Biographical_Museum	- Football	- Ancient_Greek_Theatre
	- Forest_Park	- Contemporary_Art_Museum	- Formula_One	- Cathedral
	- Hill	- Egyptian_Museum	- Handball	- Basilica
	- Historic_Park	- Erotic_Museum	- Popular_Running	- Church
	- Mountain	- Maritime_Museum	- Roller_Hockey	- Fair
	- Park	- Modern_Art_Museum	- Skateboarding	- Football_Stadium
	- Statue	- Natural_History_Museum	- Swimming	- Formula_One_Circuit
	- Urban_Park	- Science_Museum	- Swimming_Race	- Fort
	- Zoo		- Tennis	- Forum
	- Square			- Headquarter
	- Garden_Park			- Tower
	- Thematic_Park			- Hotel
	- Botanical_Park			- House
				- Industrial_Building
				- Luxury_Hotel
				- Market
				- Opera
				- Shopping
				- Skyscraper
				- Stadium
				- Theatre
				- University
				- Tower
				- Commercial_Complex
				- Business_School

Figura 71 Resultats obtinguts durant l'anàlisi

De totes les dades anteriors, és interessant comentar les que estan en vermell i explicar alguns dels motius pels quals el sistema ha determinat que eren correctes. Per exemple, en la primera ciutat s'ha considerat com a bona la paraula *Pyramid* quan en realitat al text es nombrava la piràmide per posar un exemple temporal –*about 1500 years before either Stonehenge or The Egyptian Great Pyramid of Giza was completed*–. A la mateixa ciutat, dins la secció *Sports*, apareix la paraula *Climbing* i, aparentment, es podria pensar que és correcte afegir-la a les dades trobades, però en aquest context es tracta d'un verb amb un significat diferent al de l'esport del mateix nom –*Cardiff City have played in the English Football League since the 1920–21 season, climbing to Division 1 after one season*–. A la ciutat de Bath, s'afegeix *Fair*, però en l'article no es parla d'una fira, sinó del nom d'una pel·lícula –*Many films and television programmes have been filmed in Bath including: the 2004 film of Thackeray's Vanity Fair*–.

Pel que fa a les paraules escrites en blau o verd, els principals problemes radiquen en el fet que es troben mal escrites al text (*Comercial*, en comptes de *Commercial*, *Arquitecture* en lloc d'*Architecture*...), que les paraules compostes estan separades per alguna altra (*Roman Catholic Cathedral*, *Suburban retail park*...), que en el text només es mostra el nom propi de l'objecte buscat (*Welsh National War Memorial*, que representa un *Memorial_Landmark*), que les paraules estan escrites en un altre idioma (*Vélodrome*, en francès), que no s'analitza la secció correcta de la Wikipedia on es troba la paraula buscada o que no s'interpreta el contingut del text (*The cathedral, also known as basilica La Seu is said*...). Tots aquests errors són deguts a que l'analitzador no "enten" el contingut del text.

A continuació es mostra un resum de les dades obtingudes (Figura 72). Apareixen quatre taules, una per a cada ciutat, on les files representen el nombre de conceptes trobats i no trobats i les columnes, quins d'aquests són bons i quins, dolents. A més a més, es calculen les mesures de *precision*, *recall* i *F-measure*.

Finalment, en la Figura 73 s'ha fet una comparativa entre els valors de les mesures realitzades per a cada ciutat. Es pot veure com el *recall* és lleugerament superior a la *precision* i que en totes les ciutats s'han aconseguit valors més o menys semblats, els quals oscil·len entre el 85% i 95%. Segons la mitja aritmètica calculada de la *F-measure* de totes les ciutats, el sistema té un rendiment aproximat del **92,8%**.

Barcelona

	Bones	Dolentes	Total
Trobades	50	2	52
No Trobades	7	0	7
Total	57	2	59
<i>Precision (%)</i>			87,7
<i>Recall (%)</i>			96,2
<i>F-Measure (%)</i>			91,7

Bath, Somerset

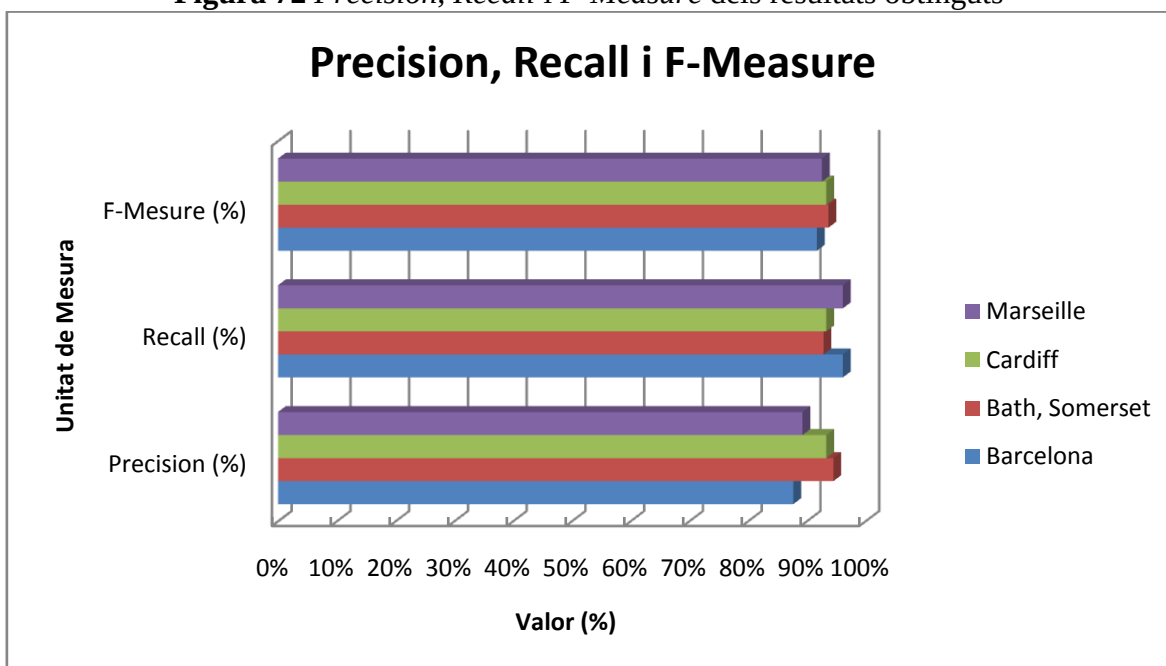
	Bones	Dolentes	Total
Trobades	52	4	56
No Trobades	3	0	3
Total	55	4	59
<i>Precision (%)</i>			94,5
<i>Recall (%)</i>			92,9
<i>F-Measure (%)</i>			93,7

Cardiff

	Bones	Dolentes	Total
Trobades	56	4	60
No Trobades	4	0	4
Total	60	4	64
<i>Precision (%)</i>			93,3
<i>Recall (%)</i>			93,3
<i>F-Measure (%)</i>			93,3

Marseille

	Bones	Dolentes	Total
Trobades	50	2	52
No Trobades	6	0	6
Total	56	2	58
<i>Precision (%)</i>			89,3
<i>Recall (%)</i>			96,2
<i>F-Measure (%)</i>			92,6

TOTAL F-Measure**92,8 %****Figura 72** Precision, Recall i F-Measure dels resultats obtinguts**Figura 73** Comparació de precision, Recall i F-Measure de les quatre ciutats

5.2 Interpretació i conclusions sobre els resultats

De tots els resultats obtinguts podem deduir alguns aspectes relacionats amb l'analitzador i les seves limitacions:

- Les dades extretes no són fiables al 100%.
- Tots els errors comesos són deguts al fet que el sistema no és capaç d'entendre el text.
- La cerca semàntica que es fa amb el suport de l'ontologia millora els resultats, però es millorarien molt més mitjançant l'anàlisi semàntica del contingut examinat.
- Els sinònims podrien aportar nous resultats.
- Els errors humans són inevitables i presents en tot moment.
- El fet que l'analitzador només compari *strings* és una limitació important.
- Els noms propis no tenen perquè aportar informació sobre l'objecte que representen.
- La diversitat d'idiomes és un impediment durant l'anàlisi.

Malgrat aquestes situacions problemàtiques, podem considerar que el sistema és bo i fiable. Respecte a la *precision*, els valors aconseguits eren els esperats, ja que aquest sistema basa la cerca en paraules de l'ontologia i no intenta deduir-ne de noves. Pel que fa al *recall*, els resultats han estat sorprenents, no s'esperaba que fossin tan elevats. Probablement, això succeeix perquè una mostra de quatre ciutats no és suficientment representativa com per extreure conclusions i perquè en els càlculs anteriors s'han considerat com a bones totes aquelles paraules marcades en blau (Figura 71).

Tot i així, es pot considerar que el sistema esdevé una bona eina i estalvia molt de treball durant el procés de recopilació i extracció de dades, encara que els resultats obtinguts han de ser supervisats per un expert dintre de l'àmbit turístic.

6 CONCLUSIONS I TREBALL FUTUR

En aquesta secció es presentaran les conclusions extretes a partir de la feina realitzada i es comentaran les línies de treball futur per ampliar i millorar el sistema analitzador.

6.1 Conclusions

Podriem dir que s'han complit els objectius d'aquest treball: s'ha definit i generat l'estructura d'una matriu atribut-valor, s'ha aconseguit una eina realment útil que ha permès crear aquestes matrius amb molta més flexibilitat de la esperada i s'ha desenvolupat un sistema de recuperació d'informació força efectiu.

El disseny d'una ontologia no és un exercici trivial. De fet, es tracta d'una tasca molt complexa i laboriosa, ja que amb ella s'intenta representar relacions i objectes del món i s'ha de saber escollir la informació adequada per a cada cas. El turisme és un àmbit molt gran i canviant, fet que ha dificultat des d'un primer moment el seu estudi. Per això, seria necessari que en la realització de l'ontologia hi hagués implicat un expert en turisme que decideixi quins conceptes haurien d'estar presents a l'ontologia.

El sistema de recuperació implementat ha mostrat bons resultats perquè s'ha utilitzat el coneixement semàntic del domini turístic, el qual ha facilitat l'extracció i integració de les dades, i, a més, s'ha aprofitat la semi-estructura de la Wikipedia.

Finalment, s'hauria de tindre en compte que, tot i els bons resultats obtinguts, seria interessant seguir treballant en el desenvolupament del sistema analitzador per intentar resoldre totes les mancances esmentades en el capítol anterior. En la següent secció es comenten algunes línies futures de treball.

6.2 Treball futur

A partir d'aquest projecte queden obertes noves línies de treball. Per tal de millorar el rendiment del sistema es podrien ampliar les fonts de informació d'on s'extreuen les dades (per exemple, es podria utilitzar Wikitravel), també es podrien utilitzar tècniques de *Stemmer* per comparar les paraules buscades, o utilitzar WordNet per obtenir conceptes sinònims de manera que es puguin ampliar els resultats trobats. S'hauria de refinar i ampliar la ontologia o, inclús, aplicar tècniques d'aprenentatge automàtic d'ontologies (*Ontology Learning*). Durant el procés d'extracció de dades es podria analitzar sintàctica i semànticament el contingut del text.

Per complir els objectius del projecte DAMASK, s'haurien d'aplicar tècniques de *clustering* per tal de classificar semànticament les dades obtingudes, dissenyar mètodes per interpretar aquestes dades i, finalment, crear un sistema de recomanació personalitzada de destins turístics per facilitar un turisme sostenible.

7 BIBLIOGRAFIA

- [1] Berners-lee, T., Hendler, J. Lassila, O.: The semantic Web. Scientific American. 2001
- [2] Borst, W.N.: Construction of Engineering Ontologies. Centre for Telematics and Information Technology, University of Twente. Enschede, The Netherlands. 1997.
- [3] Cimiano, P., Ladwig, G., Staab, S.: Gimme' the context: context-driven automatic semantic annotation with c-pankow. Actas de 14th World Wide Web Conference (2005)
- [4] D. Sánchez. Domain Ontology Learning from the Web. PhD Thesis, UPC, December 2007.
- [5] Decker, S., Erdmann, M., Fensel, D. and Suder, R.: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In Proceedings of Semantic Issues in Multimedia Systems (DS-8). Rotorua, New Zealand. 1999. 351-369
- [6] Domingos, P., Toward Knowledge-Rich Data Mining, Data Mining & Knowledge Discovery, N0. 15, pp. 21-28, 2007
- [7] E. Herrera, L., Martínez, A. 2.tupple fuzzy linguistic representation model for computing with words, IEEE Trans. On Fuzzy Systems, 8(6), 746-752, 2000
- [8] Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S. and Yates, A.: Unsupervised named-entity extraction from the Web: An experimental study. Artificial Intelligence 165. 2005. 91-134.
- [9] Fellbaum, C.: WordNet: An Electronic Lexical Database. Cambridge, Massachusetts: MIT Press. More information: <http://www.cogsci.princeton.edu/~wn/>. 1998.
- [10] Flesca, S., Manco, G., Masciari, E., Randle, E., Tagarelli, A.: Web wrapper induction: a brief survey. AI Communications 17. IOS Press. 2004. 57-61
- [11] Fox, M.S.: The TOVE Project: A Common-sense Model of the Enterprise. In Proceedings of the Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. LNAI 604. 1992. 25-34.
- [12] Gómez-Pérez, A., Fernández-López, M. and Corcho, O.: Ontological Engineering, 2nd printing. Springer Verlag.. ISBN: 1-85233-551-3. 2004
- [13] Gruber, T.R. and Olsen, F.: An ontology for Engineering Mathematics. In Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning. Bonn, Germany. 1994. 258-269.
- [14] Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. In Guarino, N., Poli, R. (eds) International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation. Padova, Italy. 1993. 907-928.
- [15] Guarino, N.: Formal Ontology in Information Systems. In Guarino N. (ed) 1st International Conference on Formal Ontology in Information Systems (FOIS'98). IOS Press. Trento, Italy. 1998. 3-15.
- [16] Hernández J., Ramírez, J., Ferri, C., Introducción a la Minería de Datos, Pearson Prentice Hall, 2005
- [17] IEEE, Special issue on the Semantic Web. IEEE Intelligent Systems. 2001.

- [18] Lassila, O., McGuinness, D.: The Role of Frame-Based Representation on the Semantic Web. Technical Report KSL-01-02. Knowledge Systems Laboratory. Stanford University. Standford, California. 2001.
- [19] Lenat, D.B. and Guha, R.V.: Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project. Addison-Wesley, Boston, Massachusetts. 1990.
- [20] Moreno, A., Riaño, D., Isern, D., Bocio, J., Sánchez, D., Jiménez, L.: Knowledge Exploitation from the Web. Procs of the 5th International Conference on Practical Aspects of Knowledge Management (PAKM 2004). LNAI, 3336. Vienna 2004. 175-185.
- [21] Murua, I., Llado, E. and Llodrá, B.: The Semantic Web for Improving Dynamic Tourist Packages Commercialisation. Leading the Web in Concurrent Engineering, IOS Press. 222-226, 2006
- [22] Neches, R., Fickes, R.E., Finin, T. Gruber, T.R., Senator, T. and Swartout W.R.: Enabling technology for knowledge sharing. AI Magazine 12(3). 1991. 36-56.
- [23] Pease, R.A. and Niles, I.: IEEE Standard Upper Ontology: A Progress Report. The Knowledge Engineering Review 17(1). 2002. 65-70.
- [24] Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove, California. 1999.
- [25] Studer, R., Benjamins, V.R. and Fensel., D.: Knowledge Engineering: Principles and Methods. IEEE Transactions on Knowledge and Data Engineering 25(1-2). 1998. 161-197.
- [26] Uschold, M. and Gruninger, M.: Ontologies. Principles, Methods and Applications. Knowledge Engineering Review 11(2), 1996. 93-155.
- [27] Uschold, M., King, M., Moralee, S. and Zorgios, Y.: The Enterprise Ontology, Knowledge Engineering Review, 13(1). 1998. 31-89.
- [28] Valls, A., Torra, V., On the semantics of qualitative attributes in knowledge elicitation, Int.J. of Intelligent Systems, 1999, vol.14, num.2, pp.195-210. ISSN: 0884-813
- [29] Van Heijst, F., Schreiber A., and Wielinga, B.J.: Using explicit ontologies in KBS development. International Journal of Human-Computer Studies 45. 1997. 183-292.
- [30] W. Graco, T.Semenova, E.Dubossarky, Toward Knowledge-Driven Data Mining, ACM SIGKDD Workshop on Domain Driven Data Mining, USA, 2007
- [31] <http://ca.wikipedia.org/wiki/Viquipèdia:Plantilles>
- [32] <http://commons.wikimedia.org/wiki/File:Mediawiki-database-schema.png>
- [33] <http://ec.grec.net/lexicx.jsp?GECART=0079587>
- [34] <http://en.wikipedia.org/wiki/Category:Catalonia>
- [35] http://en.wikipedia.org/wiki/Help:Template#Basic_template_usage_examples
- [36] <http://en.wikipedia.org/wiki/Wikipedia>
- [37] <http://en.wikipedia.org/wiki/Wikipedia:Categorization>
- [38] <http://en.wikipedia.org/wiki/Wikipedia:Edit>
- [39] http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style
- [40] <http://en.wikipedia.org/wiki/Wikipedia:MOSLINK>
- [41] http://en.wikipedia.org/wiki/Wikipedia_template
- [42] <http://es.wikipedia.org/wiki/Tesouro>
- [43] <http://e-tourism.der.at/>
- [44] <http://e-tourism.der.at/index.html>
- [45] http://members.sti2.at/~michaell/ontourism_ontology/new/

- [46] <http://upload.wikimedia.org/wikipedia/commons/4/41/Mediawiki-database-schema.png>
- [47] <http://wiki.dbpedia.org/Architecture?v=1411>
- [48] <http://www.enciclopedia.cat/cgi-bin/CercaGEC3.exe?APP=CERCAPAR&PAG=0001&PAR=metadades>
- [49] <http://www.mediawiki.org/wiki/Api>
- [50] <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>
- [51] <http://www.w3.org/TR/owl-features>
- [52] <http://www.w3.org/TR/rdf-sparql-query/>
- [53] <http://www.w3c.es/Consortio>

8 ANNEX

8.1 MediaWiki Api

```
*****
**
** This is an auto-generated MediaWiki API documentation page
**
**
** Documentation and Examples:
**
** http://www.mediawiki.org/wiki/API
**
*****

Status:      All features shown on this page should be
working, but the API
              is still in active development, and may change
at any time.
              Make sure to monitor our mailing list for any
updates.

Documentation: http://www.mediawiki.org/wiki/API
Mailing list:  http://lists.wikimedia.org/mailman/listinfo/mediawiki-api
Bugs & Requests:
http://bugzilla.wikimedia.org/buglist.cgi?component=API&bug_status=
NEW&bug_status=ASSIGNED&bug_status=REOPENED&order=bugs.delta_ts

Parameters:
format      - The format of the output
              One value: json, jsonfm, php, phpfm, wddx,
              wddxfm, xml, xmlfm, yamlfm, rawfm, txt, txtfm, dbg, dbgfm
              Default: xmlfm
action      - What action you would like to perform
              One value: sitematrix, opensearch, login,
              logout, query, expandtemplates, parse, feedwatchlist, help,
              paraminfo, purge, rollback, delete, undelete, protect, block,
              unblock, move, edit, emailuser, watch, patrol, import, userrights
              Default: help
version     - When showing help, include version for each
module
maxlag      - Maximum lag
smaxage     - Set the s-maxage header to this many seconds.
Errors are never cached
              Default: 0
maxage      - Set the max-age header to this many seconds.
Errors are never cached
              Default: 0
requestid   - Request ID to distinguish requests. This will
just be output back to you

*** ** * Modules *** ** *

* action=sitematrix (sm) *
Get Wikimedia sites list

This module requires read rights.
Example:
api.php?action=sitematrix

* action=opensearch *
This module implements OpenSearch protocol

This module requires read rights.
Parameters:
search      - Search string
limit       - Maximum amount of results to return
              No more than 100 (100 for bots) allowed.
              Default: 10
namespace   - Namespaces to search
              Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6,
              7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
              Default: 0
suggest     - Do nothing if $wgEnableMWSuggest is false
format      -
Example:
api.php?action=opensearch&search=Te

* action=login (lg) *
This module is used to login and get the authentication tokens.
In the event of a successful log-in, a cookie will be attached to
your session. In the event of a failed log-in, you will not be
able to attempt another log-in through this method for 5
seconds.
This is to prevent password guessing by automated password
crackers.

This module only accepts POST requests.
Parameters:
lgname      - User Name
lgpassword   - Password
lgdomain    - Domain (optional)
Example:
api.php?action=login&lgname=user&lgpassword=password

* action=logout *
This module is used to logout and clear session data
Example:
api.php?action=logout
```

```
* action=query *
Query API module allows applications to get needed pieces of data
from the MediaWiki databases,
and is loosely based on the old query.php interface.
All data modifications will first have to use query to acquire a
token to prevent abuse from malicious sites.

This module requires read rights.
Parameters:
titles      - A list of titles to work on
pageids     - A list of page IDs to work on
revids      - A list of revision IDs to work on
prop        - Which properties to get for the
titles/revisions/pageids
              Values (separate with '|'): info, revisions,
links, langlinks, images, imageinfo, templates, categories,
extlinks, categoryinfo, duplicatefiles
list        - Which lists to get
              Values (separate with '|'): allimages, allpages,
alllinks, allcategories, allusers, backlinks, blocks,
categorymembers, deletedrevs, embeddedin, imageusage, logevents,
recentchanges, search, usercontribs, watchlist, watchlistraw,
exturlusage, users, random, protectedtitles, globalblocks,
abuseelog, abusefilters
meta        - Which meta data to get about the site
              Values (separate with '|'): siteinfo, userinfo,
allmessages
generator    - Use the output of a list as the input for other
prop/list/meta items
              NOTE: generator parameter names must be prefixed
with a 'g', see examples.
              One value: links, images, templates, categories,
duplicatefiles, allimages, allpages, alllinks, allcategories,
backlinks, categorymembers, embeddedin, imageusage, search,
watchlist, watchlistraw, exturlusage, random, protectedtitles
redirects    - Automatically resolve redirects
indepaguids - Include an additional pageids section listing
all returned page IDs.
export       - Export the current revisions of all given or
generated pages
exportnowrap - Return the export XML without wrapping it in an
XML result (same format as Special:Export). Can only be used with
export
Examples:

api.php?action=query&prop=revisions&meta=siteinfo&titles=Main%20Pag
e&rvprop=user|comment

api.php?action=query&generator=allpages&gapprefix=API/&prop=revisio
ns
--- --- --- Query: Prop --- --- ---
--- --- ---

* prop=info (in) *
Get basic page information such as namespace, title, last touched
date, ...

This module requires read rights.
Parameters:
inprop      - Which additional properties to get:
              protection - List the protection level of
each page
talkid      - The page ID of the talk page for
each non-talk page
subjectid   - The page ID of the parent page
for each talk page
              Values (separate with '|'): protection, talkid,
subjectid, url, readable
intoken     - Request a token to perform a data-modifying
action on a page
              Values (separate with '|'): edit, delete,
protect, move, block, unblock, email, import
incontinue  - When more results are available, use this to
continue
Examples:
api.php?action=query&prop=info&titles=Main%20Page

api.php?action=query&prop=info&inprop=protection&titles=Main%20Page

* prop=revisions (rv) *
Get revision information.
This module may be used in several ways:
1) Get data about a set of pages (last revision), by setting
titles or pageids parameter.
2) Get revisions for one given page, by using titles/pageids
with start/end/limit params.
3) Get data about a set of revisions by setting their IDs with
revids parameter.
All parameters marked as (enum) may only be used with a single
page (#2).

This module requires read rights.
Parameters:
rvprop      - Which properties to get for each revision.
              Values (separate with '|'): ids, flags,
timestamp, user, size, comment, content
              Default: ids|timestamp|flags|comment|user
rvlimit     - limit how many revisions will be returned (enum)
              No more than 500 (5000 for bots) allowed.
rvstartid   - from which revision id to start enumeration
(enum)
rvendid     - stop revision enumeration on this revid (enum)
rvstart     - from which revision timestamp to start
enumeration (enum)
rvend       - enumerate up to this timestamp (enum)
rvdir       - direction of enumeration - towards "newer" or
"older" revisions
              One value: newer, older
              Default: older
rvuser      - only include revisions made by user
rvexcludeuser - exclude revisions made by user
rvexpandtemplates - expand templates in revision content
rvgeneratexml - generate XML parse tree for revision content
rvsection   - only retrieve the content of this section
rvtoken     - Which tokens to obtain for each revision
              Values (separate with '|'): rollback
rvcontinue  - When more results are available, use this to
continue
rvdiff      - Revision ID to diff each revision to.
              Use "prev", "next" and "cur" for the previous,
next and current revision respectively.
Examples:
```

```

    Get data with content for the last revision of titles "API" and
    "Main Page":

api.php?action=query&prop=revisions&titles=API|Main%20Page&rvprop=t
imestamp|user|comment|content
    Get last 5 revisions of the "Main Page":

api.php?action=query&prop=revisions&titles=Main%20Page&rvlimit=5&rv
prop=timestamp|user|comment
    Get first 5 revisions of the "Main Page":

api.php?action=query&prop=revisions&titles=Main%20Page&rvlimit=5&rv
prop=timestamp|user|comment&rvdir=newer
    Get first 5 revisions of the "Main Page" made after 2006-05-01:

api.php?action=query&prop=revisions&titles=Main%20Page&rvlimit=5&rv
prop=timestamp|user|comment&rvdir=newer&rvstart=20060501000000
    Get first 5 revisions of the "Main Page" that were not made made
    by anonymous user "127.0.0.1"

api.php?action=query&prop=revisions&titles=Main%20Page&rvlimit=5&rv
prop=timestamp|user|comment&rvexcludeuser=127.0.0.1
    Get first 5 revisions of the "Main Page" that were made by the
    user "MediaWiki default"

api.php?action=query&prop=revisions&titles=Main%20Page&rvlimit=5&rv
prop=timestamp|user|comment&rvuser=MediaWiki%20default

* prop=links (pl) *
    Returns all links from the given page(s)

This module requires read rights.
Parameters:
    plnamespace - Show links in this namespace(s) only
                  Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6,
    pllimit - How many links to return
              No more than 500 (5000 for bots) allowed.
              Default: 10
    plcontinue - When more results are available, use this to
continue
Examples:
    Get links from the [[Main Page]]:
    api.php?action=query&prop=links&titles=Main%20Page
    Get information about the link pages in the [[Main Page]]:

api.php?action=query&generator=links&titles=Main%20Page&prop=info
    Get links from the Main Page in the User and Template namespaces:

api.php?action=query&prop=links&titles=Main%20Page&plnamespace=2|10
Generator:
    This module may be used as a generator

* prop=langlinks (ll) *
    Returns all interlanguage links from the given page(s)

This module requires read rights.
Parameters:
    lllimit - How many langlinks to return
              No more than 500 (5000 for bots) allowed.
              Default: 10
    llcontinue - When more results are available, use this to
continue
Examples:
    Get interlanguage links from the [[Main Page]]:

api.php?action=query&prop=langlinks&titles=Main%20Page&redirects

* prop=images (im) *
    Returns all images contained on the given page(s)

This module requires read rights.
Parameters:
    imlimit - How many images to return
              No more than 500 (5000 for bots) allowed.
              Default: 10
    imcontinue - When more results are available, use this to
continue
Examples:
    Get a list of images used in the [[Main Page]]:
    api.php?action=query&prop=images&titles=Main%20Page
    Get information about all images used in the [[Main Page]]:

api.php?action=query&generator=images&titles=Main%20Page&prop=info
Generator:
    This module may be used as a generator

* prop=imageinfo (ii) *
    Returns image information and upload history

This module requires read rights.
Parameters:
    iiprop - What image information to get.
              Values (separate with '|'): timestamp, user,
comment, url, size, sha1, mime, metadata, archivename, bitdepth
    iilimit - How many image revisions to return
              No more than 500 (5000 for bots) allowed.
              Default: 1
    iistart - Timestamp to start listing from
    iient - Timestamp to stop listing at
    iurlwidth - If iiprop=url is set, a URL to an image scaled
to this width will be returned.
                Only the current version of the image can be
scaled.
                Default: -1
    iurlheight - Similar to iurlwidth. Cannot be used without
iurlwidth
                Default: -1
    iicontinue - When more results are available, use this to
continue
Examples:
api.php?action=query&titles=File:Albert%20Einstein%20Head.jpg&prop=
imageinfo

api.php?action=query&titles=File:Test.jpg&prop=imageinfo&iilimit=50
&iient=20071231235959&iiprop=timestamp|user|url

* prop=templates (tl) *
    Returns all templates from the given page(s)

This module requires read rights.

```

```

Parameters:
    tlnamespace - Show templates in this namespace(s) only
                  Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6,
    7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
    tllimit - How many templates to return
              No more than 500 (5000 for bots) allowed.
              Default: 10
    tlcontinue - When more results are available, use this to
continue
Examples:
    Get templates from the [[Main Page]]:
    api.php?action=query&prop=templates&titles=Main%20Page
    Get information about the template pages in the [[Main Page]]:

api.php?action=query&generator=templates&titles=Main%20Page&prop=in
fo
    Get templates from the Main Page in the User and Template
namespaces:

api.php?action=query&prop=templates&titles=Main%20Page&tlnamespace=
2|10
Generator:
    This module may be used as a generator

* prop=categories (cl) *
    List all categories the page(s) belong to

This module requires read rights.
Parameters:
    clprop - Which additional properties to get for each
category.
              Values (separate with '|'): sortkey, timestamp,
hidden
    clshow - Which kind of categories to show
              Values (separate with '|'): hidden, !hidden
    cllimit - How many categories to return
              No more than 500 (5000 for bots) allowed.
              Default: 10
    clcontinue - When more results are available, use this to
continue
    clcategories - Only list these categories. Useful for checking
whether a certain page is in a certain category
Examples:
    Get a list of categories [[Albert Einstein]] belongs to:
    api.php?action=query&prop=categories&titles=Albert%20Einstein
    Get information about all categories used in the [[Albert
Einstein]]:

api.php?action=query&generator=categories&titles=Albert%20Einstein&
prop=info
Generator:
    This module may be used as a generator

* prop=extlinks (el) *
    Returns all external urls (not interwikies) from the given
page(s)

This module requires read rights.
Parameters:
    ellimit - How many links to return
              No more than 500 (5000 for bots) allowed.
              Default: 10
    eloffset - When more results are available, use this to
continue
Examples:
    Get a list of external links on the [[Main Page]]:
    api.php?action=query&prop=extlinks&titles=Main%20Page

* prop=categoryinfo (ci) *
    Returns information about the given categories

This module requires read rights.
Parameters:
    cicontinue - When more results are available, use this to
continue
Example:
api.php?action=query&prop=categoryinfo&titles=Category:Foo|Category
:Bar

* prop=duplicatefiles (df) *
    List all files that are duplicates of the given file(s).

This module requires read rights.
Parameters:
    dflimit - How many files to return
              No more than 500 (5000 for bots) allowed.
              Default: 10
    dfcontinue - When more results are available, use this to
continue
Examples:
api.php?action=query&titles=File:Albert_Einstein_Head.jpg&prop=duplicat
efiles
api.php?action=query&generator=allimages&prop=duplicatefiles
Generator:
    This module may be used as a generator

--- --- --- --- --- Query: List --- --- --- --- ---
-- ---

* list=allimages (ai) *
    Enumerate all images sequentially

This module requires read rights.
Parameters:
    aifrom - The image title to start enumerating from.
    aiprefix - Search for all image titles that begin with this
value.
    aimsin - Limit to images with at least this many bytes
    aimaxsize - Limit to images with at most this many bytes
    ailimit - How many total images to return.
              No more than 500 (5000 for bots) allowed.
              Default: 10
    aidir - The direction in which to list
              One value: ascending, descending
              Default: ascending
    aishal - SHA1 hash of image
    aishalbase36 - SHA1 hash of image in base 36 (used in
MediaWiki)
    aiprop - Which properties to get
              Values (separate with '|'): timestamp, user,
comment, url, size, dimensions, mime, sha1, metadata, bitdepth

```

Default: timestamp|url

Examples:
Simple Use
Show a list of images starting at the letter "B"
api.php?action=query&list=allimages&aifrom=B
Using as Generator
Show info about 4 images starting at the letter "T"
api.php?action=query&generator=allimages&gailimit=4&gaifrom=T&prop=imageinfo
Generator:
This module may be used as a generator

*** list=allpages (ap) ***
Enumerate all pages sequentially in a given namespace

This module requires read rights.
Parameters:
apfrom - The page title to start enumerating from.
aprefix - Search for all page titles that begin with this value.
apnamespace - The namespace to enumerate.
One value: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
Default: 0
apfilterredir - Which pages to list.
One value: all, redirects, nonredirects
Default: all
apminsize - Limit to pages with at least this many bytes
apmaxsize - Limit to pages with at most this many bytes
apprtype - Limit to protected pages only
Values (separate with '|'): edit, move
apprlevel - The protection level (must be used with apprtype= parameter)
Can be empty, or Values (separate with '|'):
autoconfirmed, sysop
apprfiltercascade - Filter protections based on cascadingness (ignored when apprtype isn't set)
One value: cascading, noncascading, all
Default: all
aplimit - How many total pages to return.
No more than 500 (5000 for bots) allowed.
Default: 10
apdir - The direction in which to list
One value: ascending, descending
Default: ascending
apfilterlanglinks - Filter based on whether a page has langlinks
One value: withlanglinks, withoutlanglinks, all
Default: all

Examples:
Simple Use
Show a list of pages starting at the letter "B"
api.php?action=query&list=allpages&afrom=B
Using as Generator
Show info about 4 pages starting at the letter "T"
api.php?action=query&generator=allpages&gplimit=4&gapfrom=T&prop=info
Show content of first 2 non-redirect pages beginning at "Re"
api.php?action=query&generator=allpages&gplimit=2&gapfilterredir=nonredirects&gapfrom=Re&prop=revisions&rvprop=content
Generator:
This module may be used as a generator

*** list=alllinks (al) ***
Enumerate all links that point to a given namespace

This module requires read rights.
Parameters:
alcontinue - When more results are available, use this to continue.
alfrom - The page title to start enumerating from.
alprefix - Search for all page titles that begin with this value.
alunique - Only show unique links. Cannot be used with generator or prop=ids
alprop - What pieces of information to include
Values (separate with '|'): ids, title
Default: title
alnamespace - The namespace to enumerate.
One value: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
Default: 0
allimit - How many total links to return.
No more than 500 (5000 for bots) allowed.
Default: 10

Example:
api.php?action=query&list=alllinks&alunique&alfrom=B
Generator:
This module may be used as a generator

*** list=allcategories (ac) ***
Enumerate all categories

This module requires read rights.
Parameters:
acfrom - The category to start enumerating from.
acprefix - Search for all category titles that begin with this value.
acdir - Direction to sort in.
One value: ascending, descending
Default: ascending
aclimit - How many categories to return.
No more than 500 (5000 for bots) allowed.
Default: 10
acprop - Which properties to get
Values (separate with '|'): size, hidden
Default:

Examples:
api.php?action=query&list=allcategories&acprop=size
api.php?action=query&generator=allcategories&gacprefix=List&prop=info
Generator:
This module may be used as a generator

*** list=allusers (au) ***
Enumerate all registered users

This module requires read rights.
Parameters:
aufrom - The user name to start enumerating from.

auprefix - Search for all page titles that begin with this value.
augroup - Limit users to a given group name
One value: bot, sysop, bureaucrat, checkuser, steward, accountcreator, import, transwiki, ipblock-exempt, oversight, founder, rollbacker, confirmed, autoreviewer, abusefilter
auprop - What pieces of information to include.
'groups' property uses more server resources and may return fewer results than the limit.
Values (separate with '|'): blockinfo, groups, editcount, registration
aulimit - How many total user names to return.
No more than 500 (5000 for bots) allowed.
Default: 10
auwitheditonly - Only list users who have made edits

Example:
api.php?action=query&list=allusers&aufrom=Y

*** list=backlinks (bl) ***
Find all pages that link to the given page

This module requires read rights.
Parameters:
bltitle - Title to search.
blcontinue - When more results are available, use this to continue.
blnamespace - The namespace to enumerate.
Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
blfilterredir - How to filter for redirects
One value: all, redirects, nonredirects
Default: all
bllimit - How many total pages to return. If blredirect is enabled, limit applies to each level separately (which means you may get up to 2 * limit results).
No more than 500 (5000 for bots) allowed.
Default: 10
blredirect - If linking page is a redirect, find all pages that link to that redirect as well. Maximum limit is halved.
Examples:
api.php?action=query&list=backlinks&bltitle=Main%20Page
api.php?action=query&generator=backlinks&gbtitle=Main%20Page&prop=info
Generator:
This module may be used as a generator

*** list=blocks (bk) ***
List all blocked users and IP addresses.

This module requires read rights.
Parameters:
bkstart - The timestamp to start enumerating from
bkend - The timestamp to stop enumerating at
bkdir - The direction in which to enumerate
One value: newer, older
Default: older
bkids - Pipe-separated list of block IDs to list (optional)
bkusers - Pipe-separated list of users to search for (optional)
bkip - Get all blocks applying to this IP or CIDR range, including range blocks.
Cannot be used together with bkusers. CIDR ranges broader than /16 are not accepted.
bklimit - The maximum amount of blocks to list
No more than 500 (5000 for bots) allowed.
Default: 10
bkprop - Which properties to get
Values (separate with '|'): id, user, by, timestamp, expiry, reason, range, flags
Default:
id|user|by|timestamp|expiry|reason|flags
Examples:
api.php?action=query&list=blocks
api.php?action=query&list=blocks&bkusers=Alice|Bob

*** list=categorymembers (cm) ***
List all pages in a given category

This module requires read rights.
Parameters:
cmtitle - Which category to enumerate (required). Must include Category: prefix
cmprop - What pieces of information to include
Values (separate with '|'): ids, title, sortkey, timestamp
Default: ids|title
cmnamespace - Only include pages in these namespaces
Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
cmcontinue - For large categories, give the value returned from previous query
cmlimit - The maximum number of pages to return.
No more than 500 (5000 for bots) allowed.
Default: 10
cmsort - Property to sort by
One value: sortkey, timestamp
Default: sortkey
cmdir - In which direction to sort
One value: asc, desc
Default: asc
cmstart - Timestamp to start listing from. Can only be used with cmsort=timestamp
cmend - Timestamp to end listing at. Can only be used with cmsort=timestamp
cmstartsorkkey - Sortkey to start listing from. Can only be used with cmsort=sortkey
cmendsorkkey - Sortkey to end listing at. Can only be used with cmsort=sortkey
Examples:
Get first 10 pages in [[Category:Physics]]:
api.php?action=query&list=categorymembers&cmtitle=Category:Physics
Get page info about first 10 pages in [[Category:Physics]]:
api.php?action=query&generator=categorymembers&gcmtitle=Category:Physics&prop=info
Generator:
This module may be used as a generator

*** list=deletedrevs (dr) ***

List deleted revisions.
This module operates in three modes:
1) List deleted revisions for the given title(s), sorted by timestamp
2) List deleted contributions for the given user, sorted by timestamp (no titles specified)
3) List all deleted revisions in the given namespace, sorted by title and timestamp (no titles specified, druser not set)
Certain parameters only apply to some modes and are ignored in others.
For instance, a parameter marked (1) only applies to mode 1 and is ignored in modes 2 and 3.

This module requires read rights.

Parameters:

- drstart - The timestamp to start enumerating from. (1,2)
- drrend - The timestamp to stop enumerating at. (1,2)
- drdir - The direction in which to enumerate. (1,2)
One value: newer, older
Default: older
- drfrom - Start listing at this title (3)
- drcontinue - When more results are available, use this to continue (3)
- drunique - List only one revision for each page (3)
- druser - Only list revisions by this user
- drexcludeuser - Don't list revisions by this user
- drnamespace - Only list pages in this namespace (3)
One value: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
Default: 0
- drlimit - The maximum amount of revisions to list
No more than 500 (5000 for bots) allowed.
Default: 10
- drprop - Which properties to get
Values (separate with '|'): revid, user, comment, minor, len, content, token
Default: user|comment

Examples:

List the last deleted revisions of Main Page and Talk:Main Page, with content (mode 1):

```
api.php?action=query&list=deletedrevisions&titles=Main%20Page|Talk:Main%20Page&drprop=user|comment|content
```

List the last 50 deleted contributions by Bob (mode 2):

```
api.php?action=query&list=deletedrevisions&druser=Bob&drlimit=50
```

List the first 50 deleted revisions in the main namespace (mode 3):

```
api.php?action=query&list=deletedrevisions&drdir=newer&drlimit=50
```

List the first 50 deleted pages in the Talk namespace (mode 3):

```
api.php?action=query&list=deletedrevisions&drdir=newer&drlimit=50&drnamespace=1&drunique
```

*** list=embeddedin (ei) ***
Find all pages that embed (transclude) the given title

This module requires read rights.

Parameters:

- eititle - Title to search.
- eicontinue - When more results are available, use this to continue.
- einamespace - The namespace to enumerate.
Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
- EIFilterredirect - How to filter for redirects
One value: all, redirects, nonredirects
Default: all
- EIlimit - How many total pages to return.
No more than 500 (5000 for bots) allowed.
Default: 10

Examples:

```
api.php?action=query&list=embeddedin&eititle=Template:Stub
```

api.php?action=query&generator=embeddedin&eititle=Template:Stub&prop=info

Generator:
This module may be used as a generator

*** list=imageusage (iu) ***
Find all pages that use the given image title.

This module requires read rights.

Parameters:

- iuitle - Title to search.
- iucontinue - When more results are available, use this to continue.
- iunamespace - The namespace to enumerate.
Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
- IuFilterredirect - How to filter for redirects
One value: all, redirects, nonredirects
Default: all
- Iulimit - How many total pages to return. If iuredirect is enabled, limit applies to each level separately (which means you may get up to 2 * limit results).
No more than 500 (5000 for bots) allowed.
Default: 10
- Iuredirect - If linking page is a redirect, find all pages that link to that redirect as well. Maximum limit is halved.

Examples:

```
api.php?action=query&list=imageusage&iutitle=File:Albert%20Einstein%20Head.jpg
```

api.php?action=query&generator=imageusage&giutitle=File:Albert%20Einstein%20Head.jpg&prop=info

Generator:
This module may be used as a generator

*** list=logevents (le) ***
Get events from logs.

This module requires read rights.

Parameters:

- leprop - Which properties to get
Values (separate with '|'): ids, title, type, user, timestamp, comment, details
Default: ids|title|type|user|timestamp|comment|details
- letype - Filter log entries to only this type(s)
Can be empty, or One value: block, protect, rights, delete, upload, move, import, patrol, merge, suppress,

makebot, gblblock, renameuser, globalauth, gblrights, abusefilter, newusers

- lestart - The timestamp to start enumerating from.
- leend - The timestamp to end enumerating.
- ledir - In which direction to enumerate.
One value: newer, older
Default: older
- leuser - Filter entries to those made by the given user.
- letitle - Filter entries to those related to a page.
- lelimit - How many total event entries to return.
No more than 500 (5000 for bots) allowed.
Default: 10

Example:

```
api.php?action=query&list=logevents
```

*** list=recentchanges (rc) ***
Enumerate recent changes

This module requires read rights.

Parameters:

- rcstart - The timestamp to start enumerating from.
- rcend - The timestamp to end enumerating.
- rcdir - In which direction to enumerate.
One value: newer, older
Default: older
- rcnamespace - Filter log entries to only this namespace(s)
Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
- rcprop - Include additional pieces of information
Values (separate with '|'): user, comment, title, ids, sizes, redirect, patrolled, loginfo
Default: title|timestamp|ids
- rctoken - Which tokens to obtain for each change
Values (separate with '|'): patrol
- rcshow - Show only items that meet this criteria.
For example, to see only minor edits done by logged-in users, set show=minor|!anon
- !bot, anon, !anon, redirect, !redirect, patrolled, !patrolled
- rcIlimit - How many total changes to return.
No more than 500 (5000 for bots) allowed.
Default: 10
- rctype - Which types of changes to show.
Values (separate with '|'): edit, new, log

Example:

```
api.php?action=query&list=recentchanges
```

*** list=search (sr) ***
Perform a full text search

This module requires read rights.

Parameters:

- srsearch - Search for all page titles (or content) that has this value.
- srnamespace - The namespace(s) to enumerate.
Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
Default: 0
- srwhat - Search inside the text or titles.
One value: title, text
- srredirects - Include redirect pages in the search.
- sroffset - Use this value to continue paging (return by query)
Default: 0
- srlimit - How many total pages to return.
No more than 500 (5000 for bots) allowed.
Default: 10

Examples:

```
api.php?action=query&list=search&srsearch=meaning
```

```
api.php?action=query&list=search&srwhat=text&srsearch=meaning
```

```
api.php?action=query&generator=search&gsrsearch=meaning&prop=info
```

Generator:
This module may be used as a generator

*** list=usercontribs (uc) ***
Get all edits by a user

This module requires read rights.

Parameters:

- uclimit - The maximum number of contributions to return.
No more than 500 (5000 for bots) allowed.
Default: 10
- ucstart - The start timestamp to return from.
- ucend - The end timestamp to return to.
- uccontinue - When more results are available, use this to continue.
- ucuser - The user to retrieve contributions for.
- ucuserprefix - Retrieve contributions for all users whose names begin with this value. Overrides ucuser.
- ucdir - The direction to search (older or newer).
One value: newer, older
Default: older
- ucnamespace - Only list contributions in these namespaces
Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
- ucprop - Include additional pieces of information
Values (separate with '|'): ids, title, timestamp, comment, size, flags, patrolled
Default: ids|title|timestamp|comment|size|flags
- ucshow - Show only items that meet this criteria, e.g. non minor edits only: show=!minor

NOTE: if show=patrolled or show=!patrolled is set, revisions older than \$wgRCMaxAge won't be shown

Values (separate with '|'): minor, !minor, patrolled, !patrolled

Examples:

```
api.php?action=query&list=usercontribs&ucuser=YurikBot
```

```
api.php?action=query&list=usercontribs&ucuserprefix=217.121.114.
```

*** list=watchlist (wl) ***
Get all recent changes to pages in the logged in user's watchlist

This module requires read rights.

Parameters:

- wallrev - Include multiple revisions of the same page within given timeframe.
- wlstart - The timestamp to start enumerating from.
- wlend - The timestamp to end enumerating.
- wlnamespace - Filter changes to only the given namespace(s).
Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101
- wldir - In which direction to enumerate pages.
One value: newer, older

Default: older

wllimit - How many total results to return per request. No more than 500 (5000 for bots) allowed. Default: 10

wlprop - Which additional items to get (non-generator mode only).

Values (separate with '|'): ids, title, flags, user, comment, timestamp, patrol, sizes

Default: ids|title|flags

wlshow - Show only items that meet this criteria. For example, to see only minor edits done by logged-in users, set show=minor|!anon

Values (separate with '|'): minor, !minor, bot, !bot, anon, !anon, patrolled, !patrolled

Examples:

```
api.php?action=query&list=watchlist
```

api.php?action=query&list=watchlist&wlprop=ids|title|timestamp|user|comment

api.php?action=query&list=watchlist&wllallrev&wlprop=ids|title|timestamp|user|comment

api.php?action=query&generator=watchlist&prop=info

api.php?action=query&generator=watchlist&gwllallrev&prop=revisions&rwp=timestamp|user

Generator:

This module may be used as a generator

*** list=watchlistraw (wr) ***

Get all pages on the logged in user's watchlist

This module requires read rights.

Parameters:

wrcontinue - When more results are available, use this to continue

wrnamespace - Only list pages in the given namespace(s). Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101

wrlimit - How many total results to return per request. No more than 500 (5000 for bots) allowed. Default: 10

wrprop - Which additional properties to get (non-generator mode only).

Values (separate with '|'): changed

wrshow - Only list items that meet these criteria. Values (separate with '|'): changed, !changed

Examples:

```
api.php?action=query&list=watchlistraw
```

api.php?action=query&generator=watchlistraw&gwrshow=changed&prop=revisions

Generator:

This module may be used as a generator

*** list=exturlusage (eu) ***

Enumerate pages that contain a given URL

This module requires read rights.

Parameters:

euprop - What pieces of information to include. Values (separate with '|'): ids, title, url

Default: ids|title|url

euoffset - Used for paging. Use the value returned for "continue"

euprotocol - Protocol of the url. If empty and euquery set, the protocol is http.

external links - Can be empty, or One value: http, https, ftp, irc, gopher, telnet, nntp, worldwind, mailto, news, svn

Default:

euquery - Search string without protocol. See [[Special:LinkSearch]]. Leave empty to list all external links

eunamespace - The page namespace(s) to enumerate. Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101

eulimit - How many pages to return. No more than 500 (5000 for bots) allowed. Default: 10

Example:

```
api.php?action=query&list=exturlusage&euquery=www.mediawiki.org
```

Generator:

This module may be used as a generator

*** list=users (us) ***

Get information about a list of users

This module requires read rights.

Parameters:

usprop - What pieces of information to include

blockinfo - tags if the user is blocked, by whom, and for what reason

groups - lists all the groups the user belongs to

editcount - adds the user's edit count

registration - adds the user's registration

emailable - tags if the user can and wants to receive e-mail through [[Special:Emailuser]]

gender - tags the gender of the user.

Returns "male", "female", or "unknown"

Values (separate with '|'): blockinfo, groups, editcount, registration, emailable, gender

users - A list of users to obtain the same information for

ustoken - Which tokens to obtain for each user

Values (separate with '|'): userrights

Example:

```
api.php?action=query&list=users&ususers=brion|TimStarling&usprop=groups|editcount|gender
```

*** list=random (rn) ***

Get a set of random pages

NOTE: Pages are listed in a fixed sequence, only the starting point is random. This means that if, for example, "Main Page" is the first random page on your list, "List of fictional monkeys" will

always be second, "List of people on stamps of Vanuatu" third, etc.

NOTE: If the number of pages in the namespace is lower than rnlimit, you will get fewer pages. You will not get the same page twice.

This module requires read rights.

Parameters:

rnnamespace - Return pages in these namespaces only

Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101

rnlimit - Limit how many random pages will be returned

No more than 10 (20 for bots) allowed. Default: 1

rnredirect - Load a random redirect instead of a random page

Example:

```
api.php?action=query&list=random&rnnamespace=0&rnlimit=2
```

Generator:

This module may be used as a generator

*** list=protectedtitles (pt) ***

List all titles protected from creation

This module requires read rights.

Parameters:

ptnamespace - Only list titles in these namespaces

Values (separate with '|'): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100, 101

ptlevel - Only list titles with these protection levels

Values (separate with '|'): autoconfirmed, sysop

ptlimit - How many total pages to return. No more than 500 (5000 for bots) allowed. Default: 10

ptdir - The direction in which to list

One value: older, newer

Default: older

ptstart - Start listing at this protection timestamp

ptend - Stop listing at this protection timestamp

ptprop - Which properties to get

Values (separate with '|'): timestamp, user, comment, expiry, level

Default: timestamp|level

Example:

```
api.php?action=query&list=protectedtitles
```

Generator:

This module may be used as a generator

*** list=globalblocks (bg) ***

List all globally blocked IP addresses.

This module requires read rights.

Parameters:

bgstart - The timestamp to start enumerating from

bgend - The timestamp to stop enumerating at

bgdir - The direction in which to enumerate

One value: newer, older

Default: older

bgids - Pipe-separated list of block IDs to list (optional)

bgaddresses - Pipe-separated list of addresses to search for (optional)

bgip - Get all blocks applying to this IP or CIDR range, including range blocks.

Cannot be used together with bkusers. CIDR ranges broader than /16 are not accepted.

bglimit - The maximum amount of blocks to list

No more than 500 (5000 for bots) allowed. Default: 10

bgprop - Which properties to get

Values (separate with '|'): id, address, by, timestamp, expiry, reason, range

Default: id|address|by|timestamp|expiry|reason

Examples:

```
api.php?action=query&list=globalblocks
api.php?action=query&list=globalblocks&bgip=217.121.114.116
```

*** list=abuselog (af) ***

Show events that were caught by one of the abuse filters.

This module requires read rights.

Parameters:

afstart - The timestamp to start enumerating from

afend - The timestamp to stop enumerating at

afldir - The direction in which to enumerate

One value: newer, older

Default: older

afuser - Show only entries done by a given user or IP address.

aftitle - Show only entries occurring on a given page.

affilter - Show only entries that were caught by a given filter ID

afllimit - The maximum amount of entries to list

No more than 500 (5000 for bots) allowed. Default: 10

afprop - Which properties to get

Values (separate with '|'): ids, filter, user, ip, title, action, details, result, timestamp

Default: ids|user|title|action|result|timestamp

Examples:

```
api.php?action=query&list=abuselog
api.php?action=query&list=abuselog&aftitle=API
```

*** list=abusefilters (abf) ***

Show details of the abuse filters.

This module requires read rights.

Parameters:

abfstartid - The filter id to start enumerating from

abfendid - The filter id to stop enumerating at

abfdir - The direction in which to enumerate

One value: older, newer

Default: newer

abfshow - Show only filters which meet these criteria

Values (separate with '|'): enabled, !enabled, deleted, !deleted, private, !private

abflimit - The maximum number of filters to list

No more than 500 (5000 for bots) allowed. Default: 10

abfprop - Which properties to get

Values (separate with '|'): id, description, pattern, actions, hits, comments, lasteditor, lastedittime, status, private

Default: id|description|actions|status

Examples:

```
api.php?action=query&list=abusefilters&abfshow=enabled|!private
```

```

api.php?action=query&list=abusefilters&abfprop=id|description|patte
rn

---
Query: Meta
---

* meta=siteinfo (si) *
  Return general information about the site.

This module requires read rights.
Parameters:
  siprop
    - Which sysinfo properties to get:
      general      - Overall system information
      namespaces   - List of registered namespaces
and their canonical names
      namespacesaliases - List of registered namespace
aliases
      specialpagealiases - List of special page
aliases
      magicwords    - List of magic words and their
aliases
      statistics    - Returns site statistics
      interwikimap  - Returns interwiki map
(optional filtered)
      dbrepllag     - Returns database server with the
highest replication lag
      usergroups    - Returns user groups and the
associated permissions
      extensions    - Returns extensions installed on
the wiki
      fileextensions - Returns list of file
extensions allowed to be uploaded
      rightsinfo    - Returns wiki rights (license)
information if available
      languages     - Returns a list of languages
MediaWiki supports
  Values (separate with '|'): general, namespaces,
namespacesaliases, specialpagealiases, magicwords, interwikimap,
dbrepllag, statistics, usergroups, extensions, fileextensions,
rightsinfo, languages
  Default: general
  sifilteriw      - Return only local or only nonlocal entries of
the interwiki map
  One value: local, !local
  sishowalldb    - List all database servers, not just the one
lagging the most
Examples:

api.php?action=query&meta=siteinfo&siprop=general|namespaces|namesp
acealiases|statistics

api.php?action=query&meta=siteinfo&siprop=interwikimap&sifilteriw=l
ocal

api.php?action=query&meta=siteinfo&siprop=dbrepllag&sishowalldb

* meta=userinfo (ui) *
  Get information about the current user

This module requires read rights.
Parameters:
  uiprop
    - What pieces of information to include
      blockinfo    - tags if the current user is
blocked, by whom, and for what reason
      hasmsg       - adds a tag "message" if the
current user has pending messages
      groups       - lists all the groups the current
user belongs to
      rights       - lists all the rights the current
user has
      changeablegroups - lists the groups the
current user can add to and remove from
      options      - lists all preferences the current
user has set
      editcount    - adds the current user's edit
count
      ratelimits   - lists all rate limits applying to
the current user
  Values (separate with '|'): blockinfo, hasmsg,
groups, rights, changeablegroups, options, preferencestoken,
editcount, ratelimits, email
Examples:
  api.php?action=query&meta=userinfo

api.php?action=query&meta=userinfo&uiprop=blockinfo|groups|rights|h
asmsg

* meta=allmessages (am) *
  Return messages from this site.

This module requires read rights.
Parameters:
  ammessages
    - Which messages to output. "*" means all messages
  Default: *
  amfilter
    - Return only messages that contain this string
  amlang
    - Return messages in this language
  amfrom
    - Return messages starting at this message
Examples:
  api.php?action=query&meta=allmessages&amfilter=ipb-

api.php?action=query&meta=allmessages&amessages=august|mainpage&am
lang=de

*** Modules: continuation ***
***

* action=expandtemplates *
  This module expand all templates in wikitext

This module requires read rights.
Parameters:
  title
    - Title of page
    Default: API
  text
    - Wikitext to convert
  generatexml
    - Generate XML parse tree
Example:
  api.php?action=expandtemplates&text={{Project:Sandbox}}

* action=parse *
  This module parses wikitext and returns parser output

```

This module requires read rights.

Parameters:

- title - Title of page the text belongs to
 - Default: API
- text - Wikitext to parse
- page - Parse the content of this page. Cannot be used together with text and title
- redirects - If the page parameter is set to a redirect, resolve it
- oldid - Parse the content of this revision. Overrides page
- prop - Which pieces of information to get.
 - NOTE: Section tree is only generated if there are more than 4 sections, or if the `__TOC__` keyword is present
 - Values (separate with '|'): text, langlinks, categories, links, templates, images, externallinks, sections, revid, displaytitle
 - Default: text|langlinks|categories|links|templates|images|externallinks|sections|revid|displaytitle
- pst - Do a pre-save transform on the input before parsing it.
 - Ignored if page or oldid is used.
- onlypst - Do a PST on the input, but don't parse it. Returns PSTed wikitext. Ignored if page or oldid is used.

Example:

```
api.php?action=parse&text={{Project:Sandbox}}
```

*** action=feedwatchlist ***
This module returns a watchlist feed

This module requires read rights.

Parameters:

- feedformat - The format of the feed
 - One value: rss, atom
 - Default: rss
- hours - List pages modified within this many hours from now
 - The value must be between 1 and 72
 - Default: 24
- allrev - Include multiple revisions of the same page within given timeframe.

Example:

```
api.php?action=feedwatchlist
```

*** action=help ***
Display this help screen.

*** action=paraminfo ***
Obtain information about certain API parameters

Parameters:

- modules - List of module names (value of the action=parameter)
- querymodules - List of query module names (value of prop=, meta= or list= parameter)
- mainmodule - Get information about the main (top-level) module as well
- pagesetmodule - Get information about the pageset module (providing titles= and friends) as well

Example:

```
api.php?action=paraminfo&modules=parse&querymodules=allpages|siteinfo
```

*** action=purge ***
Purge the cache for the given titles.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:

- titles - A list of titles

Example:

```
api.php?action=purge&titles=Main_Page|API
```

*** action=rollback ***
Undo the last edit to the page. If the last user who edited the page made multiple edits in a row, they will all be rolled back.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:

- title - Title of the page you want to rollback.
- user - Name of the user whose edits are to be rolled back. If set incorrectly, you'll get a badtoken error.
- token - A rollback token previously retrieved through prop=revisions
- summary - Custom edit summary. If not set, default summary will be used.
- markbot - Mark the reverted edits and the revert as bot edits

Examples:

```
api.php?action=rollback&title=Main%20Page&user=Catrope&token=123ABC
```

```
api.php?action=rollback&title=Main%20Page&user=217.121.114.116&toke
n=123ABC&summary=Reverting%20vandalism&markbot=1
```

*** action=delete ***
Delete a page.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:

- title - Title of the page you want to delete. Cannot be used together with pageid
- pageid - Page ID of the page you want to delete. Cannot be used together with title
- token - A delete token previously retrieved through prop=info
- reason - Reason for the deletion. If not set, an automatically generated reason will be used.
- watch - Add the page to your watchlist
- unwatch - Remove the page from your watchlist
- oldimage - The name of the old image to delete as provided by iiprop=archivename

Examples:

```
api.php?action=delete&title=Main%20Page&token=123ABC
```

api.php?action=delete&title=Main%20Page&token=123ABC&reason=Prepari ng%20for%20move

* action=undelete *

Restore certain revisions of a deleted page. A list of deleted revisions (including timestamps) can be retrieved through list=deletedrevs

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:
title - Title of the page you want to restore.
token - An undelete token previously retrieved through
list=deletedrevs
reason - Reason for restoring (optional)
Default:
timestamps - Timestamps of the revisions to restore. If not set, all revisions will be restored.

Examples:

api.php?action=undelete&title=Main%20Page&token=123ABC&reason=Resto ring%20main%20page

api.php?action=undelete&title=Main%20Page&token=123ABC×tamps=2 0070703220045|20070702194856

* action=protect *

Change the protection level of a page.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:
title - Title of the page you want to (un)protect.
token - A protect token previously retrieved through
prop=info
protections - Pipe-separated list of protection levels, formatted action=group (e.g. edit=sysop)
expiry - Expiry timestamps. If only one timestamp is set, it'll be used for all protections.
Use 'infinite', 'indefinite' or 'never', for a neverexpiring protection.
Default: infinite
reason - Reason for (un)protecting (optional)
Default:
cascade - Enable cascading protection (i.e. protect pages included in this page)
Ignored if not all protection levels are 'sysop'
or 'protect'
watch - If set, add the page being (un)protected to your watchlist
Examples:

api.php?action=protect&title=Main%20Page&token=123ABC&protections=e dit=sysop|move=sysop&cascade&expiry=20070901163000|never

api.php?action=protect&title=Main%20Page&token=123ABC&protections=e dit=all|move=all&reason=Lifting%20restrictions

* action=block *

Block a user.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:
user - Username, IP address or IP range you want to block
token - A block token previously obtained through the gettoken parameter or prop=info
gettoken - If set, a block token will be returned, and no other action will be taken
expiry - Relative expiry time, e.g. '5 months' or '2 weeks'. If set to 'infinite', 'indefinite' or 'never', the block will never expire.
Default: never
reason - Reason for block (optional)
anononly - Block anonymous users only (i.e. disable anonymous edits for this IP)
ncreate - Prevent account creation
autoblock - Automatically block the last used IP address, and any subsequent IP addresses they try to login from
noemail - Prevent user from sending e-mail through the wiki. (Requires the "blockemail" right.)
hidename - Hide the username from the block log. (Requires the "hideuser" right.)
allowusertalk - Allow the user to edit their own talk page (depends on \$wgBlockAllowsUTEdit)
reblock - If the user is already blocked, overwrite the existing block
Examples:

api.php?action=block&user=123.5.5.12&expiry=3%20days&reason=First%2 0strike

api.php?action=block&user=Vandal&expiry=never&reason=Vandalism&nocr eate&autoblock&noemail

* action=unblock *

Unblock a user.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:
id - ID of the block you want to unblock (obtained through list=blocks). Cannot be used together with user
user - Username, IP address or IP range you want to unblock. Cannot be used together with id
token - An unblock token previously obtained through the gettoken parameter or prop=info
gettoken - If set, an unblock token will be returned, and no other action will be taken
reason - Reason for unblock (optional)
Examples:

api.php?action=unblock&id=105

api.php?action=unblock&user=Bob&reason=Sorry%20Bob

* action=move *

Move a page.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:
from - Title of the page you want to move. Cannot be used together with fromid
fromid - Page ID of the page you want to move. Cannot be used together with from.
to - Title you want to rename the page to.
token - A move token previously retrieved through
prop=info
reason - Reason for the move (optional).
movetalk - Move the talk page, if it exists.
movesubpages - Move subpages, if applicable
noredirect - Don't create a redirect
watch - Add the page and the redirect to your watchlist
unwatch - Remove the page and the redirect from your watchlist
Example:

api.php?action=move&from=Exampel&to=Example&token=123ABC&reason=Mis spelled%20title&movetalk&noredirect

* action=edit *

Create and edit pages.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:
title - Page title
section - Section number. 0 for the top section, 'new' for a new section
text - Page content
token - Edit token. You can get one of these through
prop=info
summary - Edit summary. Also section title when section=new
minor - Minor edit
notminor - Non-minor edit
bot - Mark this edit as bot
basetimestamp - Timestamp of the base revision (gotten through prop=revisions&rvprop=timestamp).
Used to detect edit conflicts; leave unset to ignore conflicts.
starttimestamp - Timestamp when you obtained the edit token.
Used to detect edit conflicts; leave unset to ignore conflicts.
recreate - Override any errors about the article having been deleted in the meantime
createonly - Don't edit the page if it exists already
ncreate - Throw an error if the page doesn't exist
captcha - Answer to the CAPTCHA
captchaid - CAPTCHA ID from previous request
watch - Add the page to your watchlist
unwatch - Remove the page from your watchlist
md5 - The MD5 hash of the text parameter, or the prependtext and appendtext parameters concatenated.
If set, the edit won't be done unless the hash is correct
prependtext - Add this text to the beginning of the page.
Overrides text.
Don't use together with section: that won't do what you expect.
appendtext - Add this text to the end of the page. Overrides text
undo - Undo this revision. Overrides text, prependtext and appendtext
undoafter - Undo all revisions from undo to this one. If not set, just undo one revision
Examples:
Edit a page (anonymous user):
api.php?action=edit&title=Test&summary=test%20summary&text=article% 20content&basetimestamp=20070824123454&token=%2B\
Prepend __NOTOC__ to a page (anonymous user):
api.php?action=edit&title=Test&summary=NOTOC&minor&prependtext=__NO TOC_%20A&basetimestamp=20070824123454&token=%2B\
Undo r13579 through r13585 with autosummary (anonymous user):
api.php?action=edit&title=Test&undo=13585&undoafter=13579&basetimes tamp=20070824123454&token=%2B\
Examples:

* action=emailuser *

Email a user.

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.

Parameters:
target - User to send email to
subject - Subject header
text - Mail body
token - A token previously acquired via prop=info
ccme - Send a copy of this mail to me
Example:
api.php?action=emailuser&target=WikiSysop&text=Content

* action=watch *

Add or remove a page from/to the current user's watchlist

This module requires read rights.
This module requires write rights.

Parameters:
title - The page to (un)watch
unwatch - If set the page will be unwatched rather than watched
Examples:
api.php?action=watch&title=Main_Page
api.php?action=watch&title=Main_Page&unwatch

* action=patrol *

Patrol a page or revision.

This module requires read rights.
This module requires write rights.

Parameters:
token - Patrol token obtained from list=recentchanges
rcid - Recentchanges ID to patrol
Example:
api.php?action=patrol&token=123abc&rcid=230672766

```

* action=import *
  Import a page from another wiki, or an XML file

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.
Parameters:
  token          - Import token obtained through prop=info
  summary         - Import summary
  xml             - Uploaded XML file
  interwikisource - For interwiki imports: wiki to import from
                  One value:
  interwiki       - For interwiki imports: page to import
  fullhistory     - For interwiki imports: import the full history,
not just the current version
  templates      - For interwiki imports: import all included
templates as well
  namespace      - For interwiki imports: import to this namespace
                  One value: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 100, 101
Examples:
  Import [[meta:Help:Parserfunctions]] to namespace 100 with full
  history:

api.php?action=import&interwikisource=meta&interwiki=Help:Parse
rFunctions&namespace=100&fullhistory&token=123ABC

* action=userrights *
  Add/remove a user to/from groups

This module requires read rights.
This module requires write rights.
This module only accepts POST requests.
Parameters:
  user          - User name
  add           - Add the user to these groups
                  Values (separate with '|'): bot, sysop,
bureaucrat, checkuser, steward, accountcreator, import, transwiki,
ipblock-exempt, oversight, founder, rollbacker, confirmed,
autoreviewer, abusefilter
  remove       - Remove the user from these groups
                  Values (separate with '|'): bot, sysop,
bureaucrat, checkuser, steward, accountcreator, import, transwiki,
ipblock-exempt, oversight, founder, rollbacker, confirmed,
autoreviewer, abusefilter
  token        - A userrights token previously retrieved through
list=users
  reason       - Reason for the change
                Default:

Example:

api.php?action=userrights&user=FooBot&add=bot&remove=sysop|bureau
crat&token=123ABC

*** ** Permissions *** **
*** **

* writeapi *
  Use of the write API
Granted to:
  all, user, bot
* apihighlimits *
  Use higher limits in API queries (Slow queries: 500 results; Fast
queries: 5000 results). The limits for slow queries also apply to
multivalue parameters.
Granted to:
  bot, sysop

*** ** Formats *** **
*** **

* format=json *
  Output data in JSON format

This module requires read rights.
Parameters:
  callback       - If specified, wraps the output into a given
function call. For safety, all user-specific data will be
restricted.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=json

* format=jsonfm *
  Output data in JSON format (pretty-print in HTML)

This module requires read rights.
Parameters:
  callback       - If specified, wraps the output into a given
function call. For safety, all user-specific data will be
restricted.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=jsonfm

* format=php *
  Output data in serialized PHP format

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=php

* format=phpfm *
  Output data in serialized PHP format (pretty-print in HTML)

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=phpfm

* format=wddx *
  Output data in WDDX format

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=wddx

* format=wddxfm *
  Output data in WDDX format (pretty-print in HTML)

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=wddxfm

* format=xml *
  Output data in XML format

This module requires read rights.
Parameters:
  xmldoublequote - If specified, double quotes all attributes and
content.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=xml

* format=xmlfm *
  Output data in XML format (pretty-print in HTML)

This module requires read rights.
Parameters:
  xmldoublequote - If specified, double quotes all attributes and
content.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=xmlfm

* format=yaml *
  Output data in YAML format

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=yaml

* format=yamlfm *
  Output data in YAML format (pretty-print in HTML)

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=yamlfm

* format=rawfm *
  Output data with the debugging elements in JSON format (pretty-
print in HTML)

This module requires read rights.
Parameters:
  callback       - If specified, wraps the output into a given
function call. For safety, all user-specific data will be
restricted.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=rawfm

* format=txt *
  Output data in PHP's print_r() format

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=txt

* format=txtfm *
  Output data in PHP's print_r() format (pretty-print in HTML)

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=txtfm

* format=dbg *
  Output data in PHP's var_export() format

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=dbg

* format=dbgfm *
  Output data in PHP's var_export() format (pretty-print in HTML)

This module requires read rights.
Example:
  api.php?action=query&meta=siteinfo&siprop=namespaces&format=dbgfm

*** Credits: ***
  API developers:
    Roan Kattouw <Firstname>.<Lastname>@home.nl (lead developer
Sep 2007-present)
    Victor Vasiliev - vasilvv at gee mail dot com
    Bryan Tong Minh - bryan . tongminh @ gmail . com
    Yuri Astrakhan <Firstname><Lastname>@gmail.com (creator,
lead developer Sep 2006-Sep 2007)

  Please send your comments, suggestions and questions to
mediawiki-api@lists.wikimedia.org
  or file a bug report at http://bugzilla.wikimedia.org/

```

8.2 TourismOWL.owl

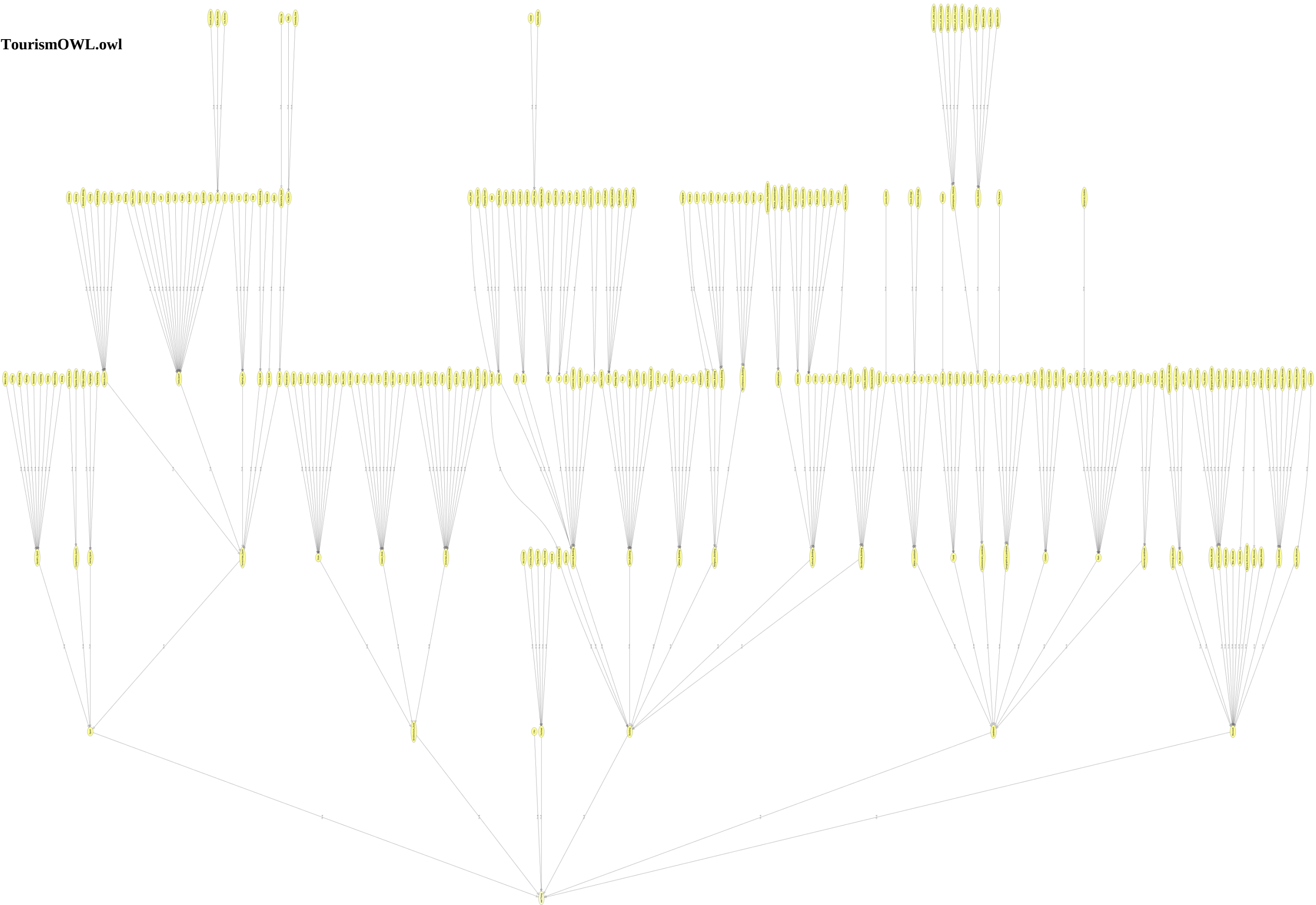


Figura 74 Ontologia turística completa

