ENRICH DATA: MILLORANT LES CERQUES A LA WEB

Autor: David Tamayo Domènech
Director: Josep Lluís Larriba Pey
Codirectors: Joan Guisado i Jordi Urmeneta

Grau en Enginyeria Informàtica
Especialització en Enginyeria del Software

UNIVERSITAT POLITÈCNICA DE CATALUNYA
Facultat d'Informàtica de Barcelona

22/06/2016

Agraïments

Vull donar les gràcies a totes aquelles persones que han participat i m'han ajudat a realitzar aquest projecte.

Primer voldria agrair a en Josep Lluís Larriba Pey l'oportunitat de realitzar aquest projecte, i sobretot l'oportunitat de formar part del grup de recerca DAMA-UPC.

També voldria agrair a tots els integrants del DAMA-UPC, que de forma directe o indirecte han contribuït al desenvolupament d'Enrich Data. En especial a en Joan Guisado i en Jordi Urmeneta per la seva ajuda i paciència, a l'Ivan de la Rubia sempre disponible per donar un cop de mà, i a l'Arnau Prat i la Dàmaris pel seu suport desinteressat.

Així mateix vull agrair a la meva família, la meva parella, Sara, i a tots els meus amics pel suport i els ànims que m'han brindat des de l'inici del projecte.

Resum

La cerca d'informació a llocs webs pot ser una tasca frustrant pels usuaris, que sovint involuntàriament utilitzen paraules clau massa generals o inapropiades per expressar les seves consultes.

Les eines de cerca per a webs actuals presenten alguns inconvenients. D'entrada, són massa genèriques i no s'adapten a les necessitats específiques de cada web. D'altra banda, requereixen uns alts recursos econòmics o tècnics, com ara un registre de consultes o motors de llenguatge natural, dels que no tothom disposa.

Les tècniques d'expansió de consultes poden ajudar a solucionar aquests problemes modificant les consultes dels usuaris, afegint-hi nous termes per tal d'obtenir millors resultats.

Per tal d'oferir una solució a aquestes necessitats s'ha dissenyat i desenvolupat Enrich Data, un servei d'expansió de consultes que s'adapta al conjunt de temes i vocabulari (domini) emprat a cada lloc web, accessible des d'un punt de vista econòmic i tècnic allotjat al núvol.

Aquest és un projecte europeu finançat per Tetracom, una iniciativa que promou la transferència tecnològica entre e l'àmbit acadèmic i l'industrial, i donarà suport a la recerca d'en Joan Guisado (codirector del projecte), que està finalitzant el seu doctorat.

Resumen

La búsqueda de información en sitios webs puede ser una tarea frustrante para los usuarios, que a menudo involuntariamente utilizan palabras clave demasiado generales o inapropiadas para expresar sus consultas.

Las herramientas de búsqueda para webs actuales presentan algunos inconvenientes. De entrada, son demasiado genéricas y no se adaptan a las necesidades específicas de cada web. Por otro lado, requieren unos altos recursos económicos o técnicos, tales como un registro de consultas o motores de lenguaje natural, de los que no todo el mundo dispone.

Las técnicas de expansión de consultas pueden ayudar a solucionar estos problemas modificando las consultas de los usuarios, añadiendo nuevos términos para obtener mejores resultados.

Para ofrecer una solución a estas necesidades se ha diseñado y desarrollado Enrich Data, un servicio de expansión de consultas que se adapta al conjunto de temas y vocabulario (dominio) empleado en cada sitio web, accesible desde un punto de vista económico y técnico alojado en la nube.

Este es un proyecto europeo financiado por Tetracom, una iniciativa que promueve la transferencia tecnológica entre el ámbito académico y el industrial, y apoyará la investigación de Juan Guisado (codirector del proyecto), que está finalizando su doctorado.

Abstract

The search for relevant information in websites can be very frustrating for users who often use involuntarily keywords too general or inappropriate to express their queries.

The existing search tools for websites have some problems. These tools are too generic and do not adapt to the specific needs of each site. On the other hand, these tools require high economic or technical resources, such as a log of queries or natural language processing engines, which not everyone has.

Query expansion techniques can help solve these problems by modifying queries adding new terms to get better results.

To supply this need, we present Enrich Data, a query expansion service that adapts to all topics and vocabulary (domain) used in each site, accessible from an economic and technical point of view hosted in the cloud.

This is an European project funded by Tetracom, an initiative that promotes technology transfer between academia and industry, and supports the research of Joan Guisado (project co-director), who is completing his PhD.

Índex

Capítol 1

Introducció

1.1 Actors implicats

En aquest apartat parlarem sobre els principals actors implicats al projecte, ja siguin actors relacionats amb el desenvolupament del projecte o bé els clients i usuaris als qui va destinat el producte.

1.1.1 Equip de disseny i desenvolupament

Aquest projecte s'ha desenvolupat dins del grup de recerca DAMA-UPC, el director del qual és en Josep Lluís Larriba, qui ha dirigit aquest projecte.
En Joan Guisado Gámez, integrant del grup de recerca que està finalitzant el seu doctorat, ha participat activament en gairebé totes les parts del projecte, donant suport sobre el disseny i el desenvolupament del projecte, i aportant els coneixements dels seus estudis relacionats amb les estructures del graf de la Wikipedia i l'expansió de consultes.
En Jordi Urmeneta, Project Manager, qui també ha participat en el disseny, el desenvolupament i l'organització del projecte.

1.1.2 Webmasters i usuaris web

Hi ha dues parts interessades que es beneficiaran d'aquest producte, per un costat els administradors de les webs (webmasters), que utilitzant el producte aconseguiran una major incidència dels seus continguts. Per un altre costat, els usuaris web que visiten les webs, ja que ells seran els que utilitzaran el sistema al fer les cerques a les webs i veuran millorada la seva experiència d'ús.

1.2 Formulació del problema

Actualment, les webs han esdevingut unes de les formes més comunes de difondre informació, i gairebé totes les institucions, persones i negocis (grans i petits) en tenen una. La cerca d'informació en un lloc web però, pot resultar un procés pesat pels usuaris que sovint es troben amb un missatge de "No s'ha trobat cap resultat". Malgrat aquest missatge, és possible que la plana web sí que contingui la informació que l'usuari està cercant, però el vocabulari que ha utilitzat per fer la cerca és diferent del que ha utilitzat l'editor de la web per generar el contingut. Per exemplificar-ho imagineu una situació en la qual un usuari cerca receptes que continguin "gallina" en un blog de cuina, però la

cerca no torna cap resultat perquè els editors de la web han fet servir el terme “pollastre” en comptes de “gallina”. Anomenarem *vocabulary mismatch* a aquest tipus d’escenaris

Així mateix, el desconeixement de l’usuari sobre els temes relacionats amb la cerca que està realitzant i el fet que no estigui familiaritzat amb els vocabularis, comporta en conseqüència que utilitzi unes paraules clau molt generals i poc concretes. Això pot comportar que no tots els continguts de la web en els que l'usuari pot estar interessat siguin retornats com a resultat de la cerca. Un altre exemple, també en el context d’un usuari que està fent una cerca en un blog d’alimentació, fa una consulta amb el terme “salsitxa” per trobar receptes amb aquest ingredient, però la cerca li retorna pocs resultats que únicament contenen el terme “salsitxa”. En canvi altres entrades al blog que contenen “hot dog” o “bratwurst” no serien retornades i podrien ser de l’interès de l’usuari. Anomenarem *topic inexperience* a aquests segons tipus d’escenaris.

Mitjançant l'expansió de les consultes es pretén cobrir aquestes necessitats.

1.3 Estat de l'art

El sistema que es vol desenvolupar, pretén millorar els resultats de les cerques a les webs mitjançant l'expansió de les consultes, afegint nou termes a aquestes. La dificultat en aquest procés d'expansió recau en seleccionar aquests termes d'expansió correctament, ja que una bona elecció millorarà els resultats de la consulta, però una inapropiada, els pot empitjorar.

S'ha de fer un incís i veure que el sistema que es desenvoluparà en aquest projecte, no s'encarrega de fer cerques a les webs, sinó de millorar els resultats dels actuals sistemes de cerques mitjançant l'expansió de les consultes.

En l'actualitat hi ha companyies especialitzades en l'emmagatzematge d'informació que han desenvolupat algunes solucions per millorar els resultats de les cerques, però proposen sistemes genèrics i poc accessibles.

Una de les solucions que actualment hi ha al mercat és Google Search Appliance[1], un sistema que integra tant hardware com software i està pensat i dissenyat per a organitzacions amb una gran capacitat econòmica, el preu ronda els 30.000 dolars. Tanmateix, per explotar el potencial d'aquest producte i obtenir bons resultats és necessari crear manualment fitxers específics sobre el vocabulari de les webs sobre les quals es vol utilitzar el producte, un procés costós i gairebé impossible de realitzar manualment per a grans webs.

També hi ha altres productes com SearchBlox[2], l'ElasticSearch[3] o l'IndexDen[4], que permeten modificar alguns paràmetres de les cerques, indexant les webs per millorar els

[1] https://www.google.com/work/search/products/gsa.html
[2] http://www.searchblox.com/
[3] https://www.elastic.co/
[4] http://indexden.com/

resultats, decidint quines pàgines o tipus de documents indexar, etc. Tots aquests productes tracten de millorar els resultats de la cerca utilitzant tècniques com la indexació del contingut de les webs, en canvi Enrich Data tracta de millorar els resultats expandint la cerca i fent que s'adapti millor al domini de la web. A més, aquests productes ofereixen un conjunt de serveis com l'anàlisi en temps real de les cerques o l'autocompletat entre d'altres, amb els que no competim. Això però, és un punt fort per al nostre servei, ja que permet als clients seguir utilitzant el motor de cerca que més els interessi, i sense perdre les seves funcionalitats, utilitzar Enrich Data per expandir les consultes millorant els seus resultats.

Enrich Data és una capa intermèdia transparent per als usuaris que visiten les webs, que enriqueix les cerques abans que aquestes arribin als motors de cerca. La capacitat d'integrar Enrich Data en webs ja implementades rau en el fet que és un servei cloud d'expansió de consultes i per tant el webmaster tan sols ha de capturar la cerca de l'usuari i enviar-la a l'API d'Enrich i utilitzar la seva resposta al cercador.

Podem concloure dient que tot i existir algunes alternatives que pretenen solucionar el problema plantejat en aquest projecte, o bé no el solucionen satisfactòriament, o es tracten d'alternatives poc accessibles econòmicament o tècnicament, i en cap cas aborden el problema seguint el nostre plantejament utilitzant l'expansió de consultes.

1.4 Estructura del document

Aquest document segueix la següent estructura. Al Capítol 2 es defineixen quins són els objectius d'aquest projecte i els possibles obstacles que ens podem trobar durant el seu transcurs. Durant el Capítol 3 es detalla la metodologia que s'ha seguit per desenvolupar aquest projecte. En el 4 es mostra la gestió econòmica. Durant el Capítol 5 es detalla la planificació temporal. Al Capítol 6 es concreten les especificacions del sistema que es desenvoluparà durant aquest projecte, junt amb els seus requisits i casos d'ús. En el Capítol 7, es dóna una visió general d'Enrich Data, on es defineixen els processos que s'executaran, l'arquitectura i les seves comunicacions. Al Capítol 8 determinem els diferents mòduls que conformen Enrich Data, explicant les funcions de cada un i com les realitzen. Al Capítol 9 es veuen els aspectes relacionats amb la sostenibilitat i el compromís social. Al Capítol 10 s'explica l'estat actual d'Enrich Data. Al 11 es pot veure el treball futur que queda per realitzar. Per finalitzar, al Capítol 12 conté les conclusions finals del treball.

Capítol 2

Objectiu i abast

L'objectiu principal del projecte és dissenyar un sistema que contribueixi en millorar les webs i alhora l'experiència dels usuaris. Per fer-ho ens proposem desenvolupar un sistema que expandeixi les consultes dels usuaris aconseguint així que obtinguin uns millors resultats.

Aquest sistema ha de ser accessible econòmicament per a la majoria de clients que vulguin usar-lo en les seves webs i no únicament enfocat a grans corporacions, també ha de ser accessible tècnicament perquè no només experts en el món del desenvolupament web el puguin utilitzar.

El sistema ha de ser modular, per obtenir la màxima flexibilitat i poder millorar els diferents mòduls en un futur o reaprofitar-los per a altres projectes, i a la vegada aconseguir un sistema robust i estable. També és imprescindible que sigui un sistema escalable per a poder atendre grans volums de consultes al mateix temps, amb la possibilitat de donar servei a una gran quantitat de clients.

Aquest projecte té com abast tractar només webs amb continguts exclusivament en anglès, com a treball futur afegiríem la funcionalitat de poder tractar altres llengües.

2.1 Objectius específics

Per assolir els objectius principals, s'han d'assolir abans uns objectius específics. Per a poder realitzar l'expansió de consultes cal:

- Ser capaç d'extreure el contingut de les webs automàticament.
- Ser capaç d'identificar les entitats rellevants del contingut de les webs.
- Generar un graf que contingui les relacions entre tots els articles i categories de la Wikipedia utilitzant la base de dades Sparksee (1).
- Ser capaç d'identificar i emmagatzemar tota aquella informació rellevant del contingut de les webs.
- Analitzar la informació de les webs identificant les entitats que estan fortament relacionades.
- Ser capaç de servir les peticions amb uns temps de resposta assumibles al generar

l’expansió de les consultes des de la perspectiva de la cerca d’informació.

Per a produir un producte accessible tècnicament cal:

- Allotjar el sistema al núvol per tal que els clients no hagin d’acoblar cap sistema a les seves màquines.
- Implementar una API senzilla i entenedora, per aconseguir que la interacció entre els clients i el sistema sigui òptima i clara, i que no requereixi un alt coneixement sobre desenvolupament web per a poder-la utilitzar correctament.

2.2 Possibles obstacles

2.2.1 Temps limitat

Al ser un treball de fi de grau, aquest està acotat a unes dates d'entrega, i per tant ens trobem amb un període de temps limitat. Com a conseqüència pot ser que alguns requisits quedin com a treball futur.

2.2.2 Mal acoblament dels diferents mòduls

Es tracta d'un sistema complex amb sis mòduls independents que es comuniquen entre ells i això pot comportar un plus de dificultat, a l'hora de realitzar el disseny, la implementació i el testeig.

2.2.3 Ús de noves tecnologies

Utilitzar eines amb les quals no estem familiaritzats i tenim una manca d’experiència, com gestors de comunicacions o altres eines per a identificar entitats en un text, pot comportar una major despesa de temps de l’estimat.

Capítol 3

Metodologia i rigor

3.1 Mètodes de treball

Gairebé tot el projecte serà desenvolupat per una sola persona, amb el suport del director i els co-directors del projecte, i es seguirà una metodologia àgil, amb *sprints* d'una sola setmana.

S'han establert dues reunions setmanals, una els dilluns de mitja hora, per veure l'estat el projecte i definir les tasques que es realitzaran durant la setmana, i una segona reunió el dijous d'una a dues hores, on també s'analitzarà l'estat de projecte, però a més es discutirà el disseny i la implementació del sistema, i on també s'analitzaran resultats de les proves i els tests.

3.2 Eines de seguiment

Al tractar-se d'un equip petit, on tots treballen en una mateixa sala hi haurà una comunicació diària, a més s'utilitzaran el mail i el Skype com eines de comunicació a distància.

S'utilitzarà el Trello, una eina de gestió de projectes, per organitzar, crear i assignar les tasques a fer.

3.3 Mètode de validació

Anirem fent tests sobre el sistema amb webs reals, i anirem analitzant els resultats durant les reunions que estan especificades en els subapartats anteriors d'aquest capítol.

En aquestes mateixes reunions també es veuran quins requisits i objectius ja hem assolit, i quins queden per completar, controlant també que els terminis establerts es van complint i reajustar la planificació si cal.

Per seguretat i comoditat també utilitzarem el SVN com a eina de control de versions, a més es preveu utilitzar JUnit per realitzar proves de regressió.

Capítol 4

Planificació temporal

El projecte tindrà una durada de 6 mesos i es desenvoluparà entre el 15 de desembre i el 15 de juny.

4.1 Definició de les tasques

Hem definit una primera tasca inicial dedicada la planificació i l'organització del projecte junt amb l'estat de l'art, més sis tasques, una per cada mòdul del sistema definit a l'abast.

Per cada una de les sis darreres tasques, hi haurà una fase inicial de disseny, seguit d'una segona fase d'implementació, i per acabar, una fase final de testeig i proves.

Les tasques s'han ordenat temporalment, segons les dependències que hi ha entre elles. Començant pel registre d'una web al sistema, fins que aquest ja està llest per a fer expansió de consultes per aquesta plana web.

- Planificació i organització: Per arribar a bon port, ha d'haver-hi una bona planificació i distribució de les tasques restants. Per tant, aquesta és la primera tasca, ja que tot el projecte depèn d'aquesta.
- Disseny i implementació del Coordinador: El mòdul que s'implementa en aquesta tasca és el que s'encarregarà de sincronitzar la resta dels mòduls, i interactuar amb els clients, sense aquest mòdul, la resta no serien funcionals
- Disseny i implementació de l'Extractor de contingut: Quan una web s'ha registrat al sistema el primer pas per poder processar el seu contingut serà emmagatzemar aquest, aquesta funció la realitza aquest mòdul.
- Disseny i implementació de l'Extractor d'entitats: Una vegada la tasca tres ha estat realitzada, necessitem l'extractor d'entitats, ja que calen les entitats d'una web per poder generar la seva Knowledge Base (KB).
- Disseny i implementació del Generador de la KB: Amb les entitats d'una web ja extretes, el següent pas és generar la KB.
- Disseny i implementació del Cercador de relacions: Abans de poder fer les expansions de les consultes d'una web, primer cal extreure les relacions que hi ha a seva KB.

- Disseny i implementació del Query expander: Aquesta és l'última tasca, un cop aquesta estigui enllestida, el projecte ja estarà preparat per començar a realitzar l'expansió de les consultes.

4.2 Recursos

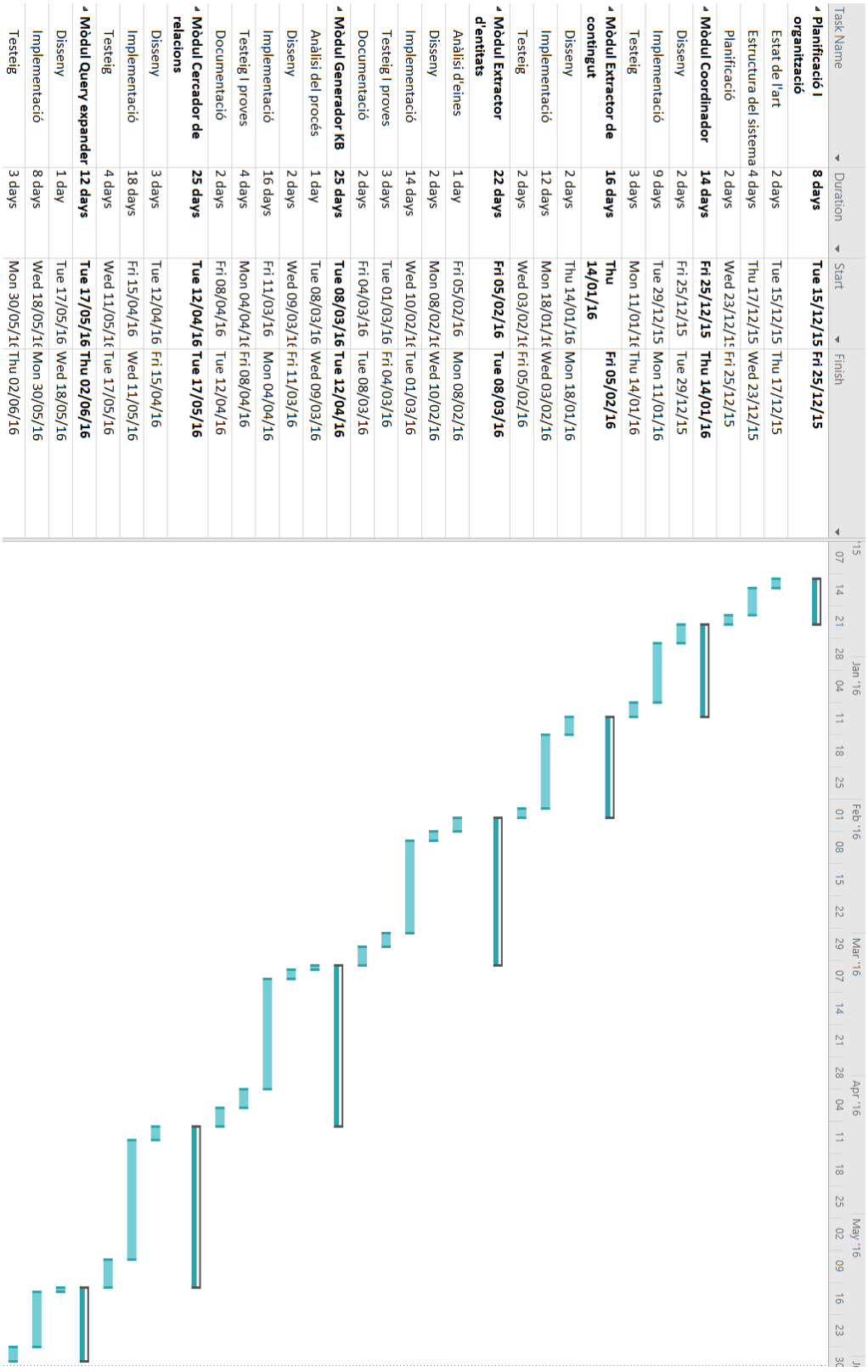
Els principals recursos que s'utilitzaran durant el projecte són:

- Oficines del grup de recerca DAMA-UPC, a l'edifici C6, campus nord de la UPC.
- Portàtil Lenovo amb Linux, amb el que principalment es desenvoluparà el sistema.
- Servidor Linux, el qual s'utilitzarà per a les execucions dels processos amb costos alts de càlcul.
- Servidor exterior: on s'aniran acoblant els diferents mòduls del projecte.
- Servidors amb els repositoris svn.
- Software varis, com Eclipse, SublimeText, etc, i altres eines com les bases de dades Sparksee i MongoDB (2).

4.2.1 Recursos humans

Els recursos humans destinats a aquest projecte, són els ja definits als actors implicats. Un equip de 4 integrants, el director del projecte, dos codirectors i un becari, aquest últim és qui tindrà la càrrega de treball més significativa al projecte, sempre amb el suport i l'ajuda de la resta d'integrants de l'equip.

4.3 Diagrama de Gantt



Task Name	Duration	Start	Finish
Planificació I organització	**8 days**	**Tue 15/12/15**	**Fri 25/12/15**
Estat de l'art	2 days	Tue 15/12/15	Thu 17/12/15
Estructura del sistema	4 days	Thu 17/12/15	Wed 23/12/15
Planificació	2 days	Wed 23/12/15	Fri 25/12/15
Mòdul Coordinador	**14 days**	**Fri 25/12/15**	**Thu 14/01/16**
Disseny	2 days	Fri 25/12/15	Tue 29/12/15
Implementació	9 days	Tue 29/12/15	Mon 11/01/16
Testeig	3 days	Mon 11/01/16	Thu 14/01/16
Mòdul Extractor de contingut	**16 days**	**Thu 14/01/16**	**Fri 05/02/16**
Disseny	2 days	Thu 14/01/16	Mon 18/01/16
Implementació	12 days	Mon 18/01/16	Wed 03/02/16
Testeig	2 days	Wed 03/02/16	Fri 05/02/16
Mòdul Extractor d'entitats	**22 days**	**Fri 05/02/16**	**Tue 08/03/16**
Anàlisi d'eines	1 day	Fri 05/02/16	Mon 08/02/16
Disseny	2 days	Mon 08/02/16	Wed 10/02/16
Implementació	14 days	Wed 10/02/16	Tue 01/03/16
Testeig I proves	3 days	Tue 01/03/16	Fri 04/03/16
Documentació	2 days	Fri 04/03/16	Tue 08/03/16
Mòdul Generador KB	**25 days**	**Tue 08/03/16**	**Tue 12/04/16**
Anàlisi del procés	1 day	Tue 08/03/16	Wed 09/03/16
Disseny	2 days	Wed 09/03/16	Fri 11/03/16
Implementació	16 days	Fri 11/03/16	Mon 04/04/16
Testeig I proves	4 days	Mon 04/04/16	Fri 08/04/16
Documentació	2 days	Fri 08/04/16	Tue 12/04/16
Mòdul Cercador de relacions	**25 days**	**Tue 12/04/16**	**Tue 17/05/16**
Disseny	3 days	Tue 12/04/16	Fri 15/04/16
Implementació	18 days	Fri 15/04/16	Wed 11/05/16
Testeig	4 days	Wed 11/05/16	Tue 17/05/16
Mòdul Query expander	**12 days**	**Tue 17/05/16**	**Thu 02/06/16**
Disseny	1 day	Tue 17/05/16	Wed 18/05/16
Implementació	8 days	Wed 18/05/16	Mon 30/05/16
Testeig	3 days	Mon 30/05/16	Thu 02/06/16

Figura 1 Diagrama de Gantt

4.4 Pla d'acció i alternatives

Com es pot veure al diagrama de Gantt, s'ha deixat un marge de temps per a possibles inconvenients.

Al seguir una metodologia àgil, amb dues reunions setmanals, els possibles errors en les estimacions de dates, o els endarreriments respecte a la planificació seran detectats ràpidament, i d'aquesta forma es podrà replantejar i reorganitzar la planificació.

El pla d'acció sobre possibles desajustaments de temps no previstos i endarreriments, consistirà primerament a tractar de reorganitzar i plantejar de quina manera es pot recuperar el ritme adequat sense utilitzar hores del marge final, ja que aquest marge ens pot fer falta més endavant. Si no hi ha una altra opció, s' assignaran hores d'aquest marge a les tasques enrederides.

Capítol 5

Gestió econòmica

5.1 Consideracions inicials

En el present capítol s'estudiaran els recursos tant humans com materials, i es calcularan els costos seguint dos plantejaments. Primerament, plantejant el projecte com un treball de final de grau, i per tant amb uns costos de recursos humans més baixos que els que hi ha actualment al mercat, i, en segon lloc, en un context on el projecte estaria situat en el mercat actual, amb els costos que això implica.

5.2 Identificació i estimació de costos

5.2.1 Costos directes per activitat

Els costos directes per activitat, engloben aquells costos relacionats amb els recursos humans que participen en el projecte.

Hi ha quatre persones directament relacionades amb el disseny i desenvolupament del projecte. El director, participant en la direcció general del projecte, dos codirectors que assistiran i participaran en totes les taques del projecte, i un becari, qui s'encarregarà en gran part del desenvolupament del projecte.

El projecte té una durada de sis mesos, i es seguirà una metodologia àgil, amb dues reunions a la setmana.

Els codirectors li dedicaran al projecte 5 hores setmanals, mentre que el director li dedicarà 2 hores a la setmana, i per últim el becari, qui li dedicarà 25 hores setmanals, durant tota la durada del projecte. Per tant, les tasques del projecte no es repartiran entre els 4 implicats, sinó que tots participaran setmanalment en aquestes.

Donat això, no té sentit fer una estimació de costos segons la repartició de tasques, ja que desglossar cada una d'aquestes per a cada un dels 4 integrants no seria útil.

Així que farem el càlcul segons les hores totals, pensant en les hores setmanals que dedica cada integrant de l'equip.

Rol	Hores	Preu real	Preu mercat	Cost real	Cost mercat
Director del projecte	50 h	25 €/h	35 €/h	1.250 €	1.750 €
Codirector 1	125 h	15 €/h	25 €/h	1.875 €	3.125 €
Codirector 2	125 h	15 €/h	25 €/h	1.875 €	3.125 €
Becari	625 h	7,5 €/h	15 €/h	4.687 €	9.375 €
Total	925 h			9.687 €	17.375 €

5.2.2 Costos indirectes

Els únics costos indirectes que ens trobem en aquest projecte és el hardward utilitzat durant el desenvolupament d'aquest, que principalment està format per un portàtil Lenovo, un servidor Linux allotjat a la universitat, i un servidor extern de pagament..

No s'utilitzarà cap software de pagament, ni tampoc es té en compte instal·lacions, llum i internet perquè les oficines són de la UPC.

Producte	Preu	Unitats	Vida útil	Temps d'ús	Preu per mes	Cost
Portàtil	1000 €	1	5 anys	6 mesos	17 €/mes	102 €
Servidor	4000 €	1	6 anys	6 mesos	56 €/mes	336 €
Servidor extern	-	1	-	3 mesos	50 €/mes	150 €
Total						588 €

5.2.3 Contingència

Prendrem un valor del 15% de contingència per a despeses no previsibles.

Costos	Percentatge	Preu mercat	Cost mercat	Preu real	Cost real
Costos directes	15%	17.375 €	2.607 €	9.687 €	1.454 €
Costos indirectes	15%	588 €	88,2 €	588 €	88,2 €
Total			2.695 €		1.544 €

5.2.4 Imprevistos

- Averies de hardware: en aquest apartat valorem possibles problemes amb el portàtil o el servidor local. Assignem una probabilitat del 5% a cada una de les

dues màquines.

- Retard de dues setmanes: Com s'ha explicat a la metodologia i al pla d'acció, cada setmana es realitzaran dues reunions, per veure l'estat del projecte i ajustar possibles desviaments de temps. Per aquest motiu, marquem amb un 10% la probabilitat que el projecte tingui un retard de dues setmanes respecte al temps estimat.

Imprevist	Percentatge	Unitats	Preu mercat	Cost mercat	Preu real	Cost real
Retard de dues set.	10%	64 h	15 €/h	960 €	7,5 €/h	480 €
Averia portàtil	5%	1	1000 €	50 €	438 €	50 €
Averia servidor	5%	1	4000 €	200 €	4000 €	200 €
Total				1.210 €		730 €

Assumim que el becari es faria càrrec de totes les hores de retard.

5.2.5 Pressupost

Cost final:

Concepte	Cost mercat	Cost real
Costos directes	17.375 €	9.687 €
Costos indirectes	588 €	588 €
Contingència	2.695 €	1.544 €
Imprevistos	1.210 €	730 €
Total	21.868 €	12.549 €

5.3 Control de gestió

Al ser un projecte amb un equip de persones petit, i amb uns recursos materials reduïts, la gestió dels recursos no serà molt complexa.

Els costos utilitzats es controlaran al final de la realització de cada mòdul, per veure si ens estem desviant dels costos previstos, i en cas que hi hagi una desviació, aplicar

mesures per corregir-la junt amb un anàlisi per veure que ha produït el desajustament.

Si la desviació no pogués ser reajustada, i la conseqüència d'aquesta augmentés els costos, aquests s'assumiran agafant fons del marge de contingència. Aquesta mesura però, sempre serà l'últim recurs.

Capítol 6

Especificació

En aquest capítol es definiran els requisits del sistema junt amb els casos d'ús.

6.1 Requisits no funcionals

Com ja s'ha explicat anteriorment, tota la interacció entre el sistema i l'usuari es fa mitjançant una API, aquesta està documentada a la plana web del projecte. L'API ha de ser senzilla, i la documentació ha de ser clara i entenedora, per facilitar l'accessibilitat al sistema.

Els clients han de disposar d'internet per fer el registre de les seves webs al sistema, qualsevol navegador o un terminal seria suficient, ja que la interacció s'efectua amb crides HTTP. També han de tenir un servidor, on allotjar les seves web i des del que faran les crides d'expansió de consultes.

El sistema s'ha implementat amb Java i Go (3), i per tant el servidor que l'allotgi pot ser Linux o Windows. També ha de tenir accés a internet per rebre i respondre les peticions dels clients.

Tot el projecte s'ha desenvolupat amb el IDE Eclipse[5]. Cinc dels sis mòduls s'han implementat amb Java, en els que hem utilitzat Maven (4) per gestionar les dependències. S'ha utilitzat el *plug-in* GoEclipse[6] de Eclipse per implementar el mòdul que s'encarrega de rebre les peticions dels clients i sincronitza la resta dels mòduls, que s'ha implementat amb Go, amb el framework Revel[7].

6.2 Requisits funcionals

Els casos d'ús s'encarreguen de representar com els diferents actors interactuaran amb el sistema.

Amb els casos d'ús veurem les seqüencies d'esdeveniments que van succeint entre el sistema i l'usuari com a conseqüència d'una acció produïda per un d'aquests dos.

[5] https://eclipse.org/
[6] http://goclipse.github.io/
[7] https://revel.github.io/

6.2.1 Diagrama de casos d'ús

En aquest projecte podem identificar tres actors diferents:

- Els webmasters: administren les webs que són registrades al sistema.
- Administrador d'Enrich Data: s'encarrega de gestionar el sistema i solucionar possibles problemes.
- Usuaris de les webs: visiten les webs registrades al sistema.

Seguidament presentarem els diagrames dels diferents actors.

Podem veure el diagrama de casos d'ús del webmaster a la Figura 2.

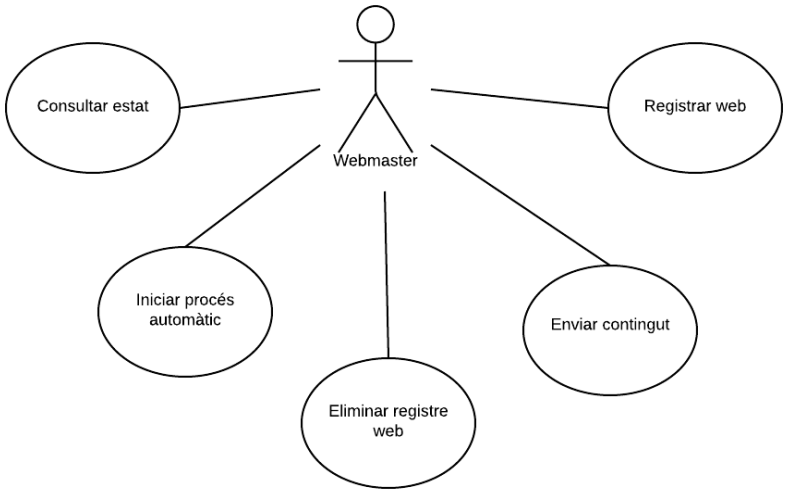


Figura 2 Diagrama de casos d'ús del webmaster

Podem veure el diagrama de casos d'ús de l'administrador d'Enrich Data a la Figura 3.

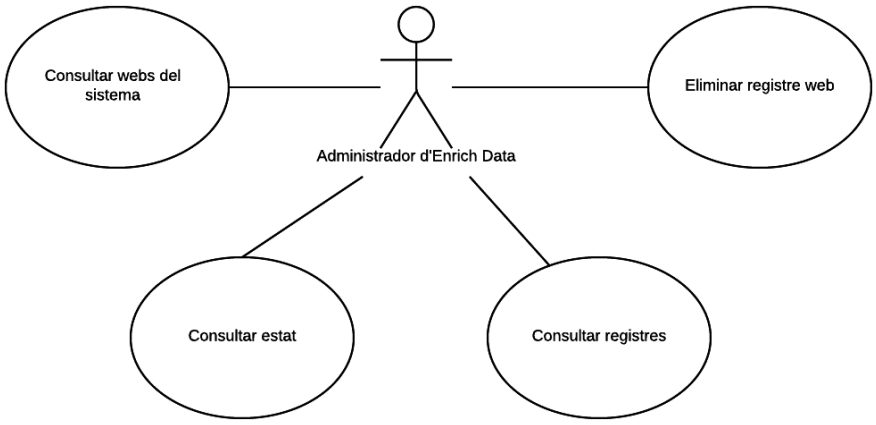


Figura 3 Diagrama de casos d'ús de l'administrador d'Enrich Data

Podem veure el diagrama de casos d'ús dels usuaris que visiten les webs a la Figura 4.

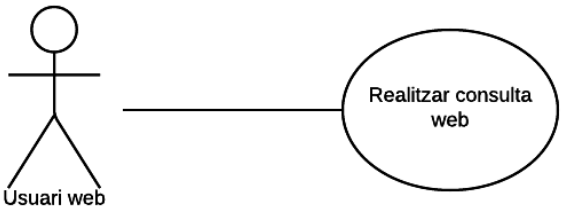


Figura 4 Diagrama de casos d'ús dels usuaris web

6.2.2 Descripció de casos d'ús

Per a cada cas d'ús, es definirà quins actors estan involucrats, i el curs d'esdeveniments entre els actors i el sistema.

Cas d'ús	Registrar web.
Resum	Permet que un usuari registri una web al sistema.
Actor	Webmaster.

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un webmaster vol registrar una web al sistema. 2. Fa la crida a l'API per registrar la web, indicant la url de la web, i una contrasenya.	
	3. El sistema registra la web al sistema, i retorna al webmaster l'identificador del web al sistema.
Curs alternatiu	
	3. La web ja ha estat registrada amb anterioritat.

Cas d'ús	Consultar estat.
Resum	Permet consultar l'estat en el qual es troba el sistema per una web.
Actor	Webmaster i administrador d'Enrich Data.

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un webmaster o administrador d'Enrich Data decideix consultar l'estat del servei per una web. 2. Fa la crida a l'API per consultar l'estat, indicant l'identificador de la web i la contrasenya.	
	3. El sistema retorna a l'actor una resposta http que conté un json amb la descripció de l'estat.
Curs alternatiu	
	3. O bé la web no està registrada, o la contrasenya és incorrecte, i es retorna un missatge d'error a l'usuari.

Cas d'ús	Enviar contingut.
Resum	Permet enviar manualment el contingut d'una web al sistema.
Actor	Webmaster i administrador d'Enrich Data.

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un webmaster o administrador decideix enviar manualment el contingut d'una web ja registrada al sistema 2. Fa la crida a l'API per enviar el contingut, indicant l'identificador de la web i la contrasenya, junt amb el contingut de la web	
	3. El sistema emmagatzema el contingut i retorna un missatge d'operació completada correctament a l'usuari
Curs alternatiu	
	3. O bé la web no està registrada, o la contrasenya és incorrecte, i es retorna un missatge d'error a l'usuari.

Cas d'ús	Eliminar registre web.
Resum	Permet donar de baixa una web ja registrada al sistema.
Actor	Webmaster i administrador d'Enrich Data

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un webmaster o administrador decideix donar de baixa una web ja registrada al sistema. 2. Fa la crida a l'API per donar de baixa la web, indicant l'identificador de la web i la contrasenya.	
	3. El sistema elimina totes les dades de la web emmagatzemades a les bases de dades relacionades amb la web, i retorna un missatge indicant que l'operació ha estat completada correctament a l'usuari.
Curs alternatiu	
	3. O bé la web no està registrada, o la contrasenya és incorrecte, i es retorna un missatge d'error a l'usuari.

Cas d'ús	Realitzar consulta web.
Resum	Permet l'expansió de consultes.
Actor	Usuari web.

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari web que està visitant una web fa una cerca a la web. 2. El servidor que allotja la web, fa una crida a l'API, indicant la consulta, l'identificador de la web i la contrasenya.	
	3. El sistema fa l'expansió de la consulta, i li retorna el resultat al servidor de la web.

4. El servidor rep la consulta expandida.	
Curs alternatiu	
	3. O bé la web no està registrada, la contrasenya és incorrecte o el servei encara no està llest per fer l'expansió de consultes, i es retorna un missatge d'error a l'usuari.

Cas d'ús	Iniciar procés automàtic.
Resum	Permet iniciar el procés automàtic de processament d'una web per poder realitzar les expansions de consultes.
Actor	Webmaster

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un webmaster decideix iniciar el procés automàtic de processament d'una web ja registrada al sistema. 2. Fa la crida a l'API indicant l'identificador de la web i la contrasenya.	
	3. El sistema inicia el procés, i retorna un missatge indicant que el procés s'ha iniciat correctament.
Curs alternatiu	
	3. O bé la web no està registrada, o la contrasenya és incorrecte, i es retorna un missatge d'error.

Cas d'ús	Consultar webs del sistema.
Resum	Permet consultar les registrades al sistema.
Actor	Administrador d'Enrich Data.

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un administrador d’Enrich Data vol consultar quines webs estan registrades. 2. Fa la crida a l'API indicant per consultar les webs registrades.	
	3. El sistema li retorna a l'administrador les webs al registrades.

Cas d'ús	Consultar registres.
Resum	Permet consultar els registres que s'han realitzat sobre una web.
Actor	Administrador d’Enrich Data.

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un administrador d’Enrich Data vol consultar els registres emmagatzemats sobre una web. 2. Fa la crida a l'API indicant l’identificador de la web i la contrasenya.	
	3. El sistema li retorna el conjunt de registres emmagatzemats sobre la web indicada a la crida.
Curs alternatiu	
	3. O bé la web no està registrada, o la contrasenya és incorrecte, i es retorna un missatge d'error a l'usuari.

Capítol 7

Visió general

En aquest capítol veurem quina és l'arquitectura d'Enrich Data, i quins mòduls la composen així com la seqüencia de processos que es realitzen des que una web es registra a Enrich Data fins que el sistema ja està llest per realitzar l'expansió de consultes.

7.1 Arquitectura general

Podem veure l'estructura general del sistema a la Figura 5:

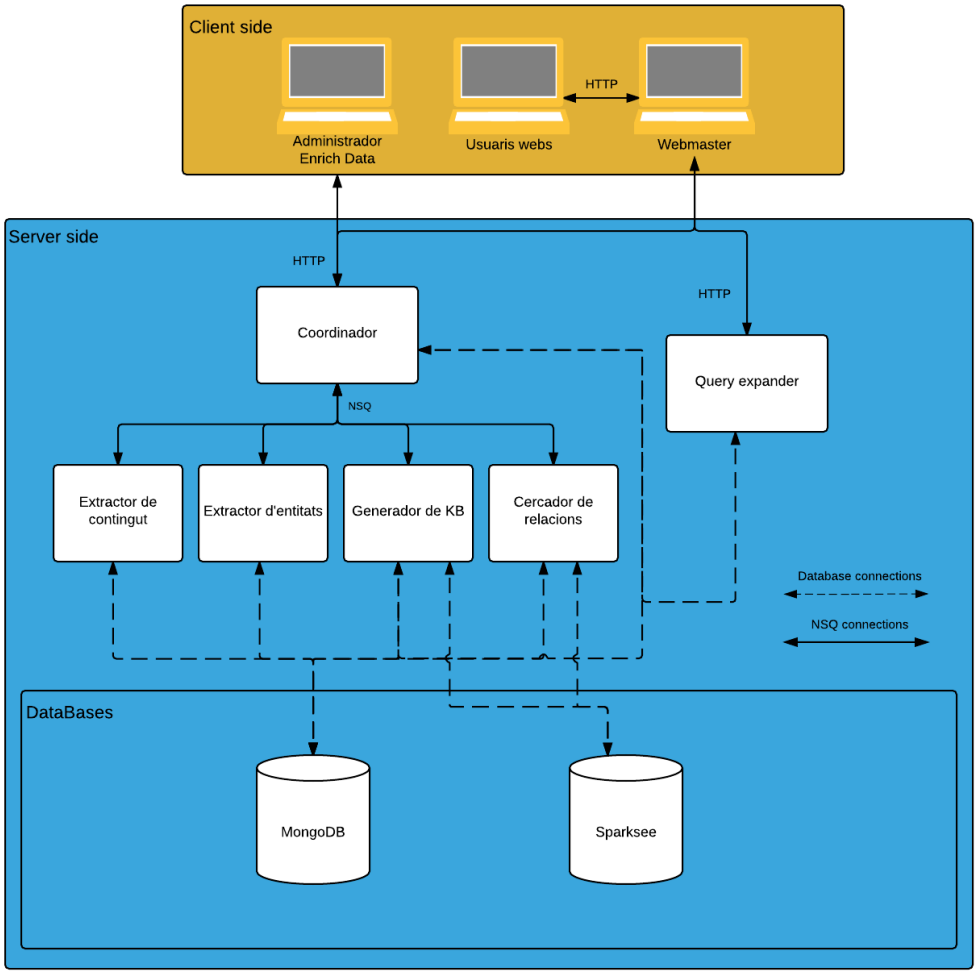


Figura 5 Arquitectura general d'Enrich Data.

Podem observar dues parts diferenciades a l'arquitectura d'Enrich Data, el client side i el server side.

Al client side ens trobem amb els diferents usuaris que interactuen en el sistema:

- Webmaster: que farà crides HTTP al servidor per registrar web i poder gestionar-les.
- Administrador d'Enrich Data: que farà peticions al sistema per controlar l'estat del sistema i poder administrar-lo.
- Usuaris webs: aquests usuaris no interactuen directament amb Enrich Data, ja que les consultes que fan a les webs van al servidor que allotja la web, que seguidament fa la petició d'expansió a Enrich Data.

Al server side trobem l'estructura d'Enrich Data. El sistema està dividit en 6 mòduls, un dels quals és el mòdul Coordinador, que s'encarrega de sincronitzar-los comunicant-se amb tots ells, amb l'excepció del mòdul Query expander que es troba aïllat de la resta. Els dos mòduls que hem esmentat anteriorment, són els únics que interactuen directament amb els webmasters i els administradors d'Enrich Data.

També podem veure que el sistema utilitza dues bases de dades diferents:

- Sparksee una base de dades orientada a grafs, que emmagatzema multigrafs dirigits, amb etiquetat i atributs.
- MongoDB una base de dades orientada a documents. MongoDB emmagatzema les dades en documents de tipus JSON. Els elements que s'emmagatzemen es denominen documents, i es guarden en col·leccions. Una col·lecció pot tenir un nombre indeterminat de documents.

7.2 Seqüencia de processos

El pipeline del sistema és seqüencial, és a dir, els processos s'executen en ordre i cadascun espera que l'anterior finalitzi per començar. A la Figura 6 podem veure la seqüència dels processos que es realitzen a Enrich Data, des que una web es registrada fins que Enrich Data està llest per a realitzar les expansions de les consultes.

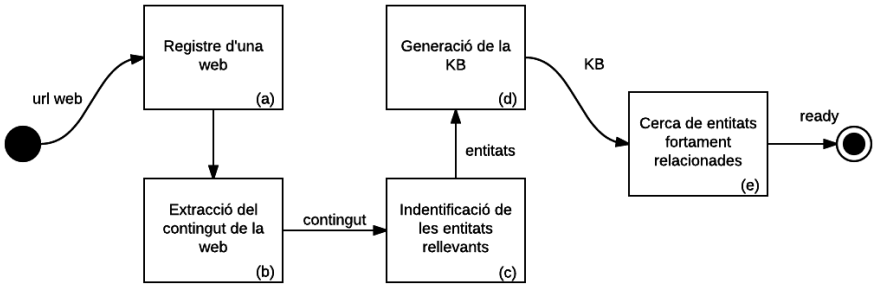


Figura 6 Seqüencia de processos d'Enrich Data.

El primer que cal és que un webmaster decideixi registrar una web a Enrich Data (a). Això inicia el primer procés d'una cadena de processos, que comença per extreure i emmagatzemar el contingut de la web en una base de dades (b). D'aquest primer procés se n'encarrega el mòdul Extractor de contingut.

Una vegada tot el contingut s'ha emmagatzemat, s'inicia un segon procés (c) executat pel mòdul Extractor d'entitats, que identifica les entitats més rellevants del contingut emmagatzemat, més endavant a l'apartat on definim aquest mòdul veurem amb més detall què són les entitats i com les identifiquem.

Seguidament, el mòdul Generador de la KB, genera una KB específica (d) per a la web a partir de la Wikipedia i les entitats extretes de la web.

Amb la KB generada, el mòdul Extractor de coneixement realitza un últim procés (e), que s'encarrega d'analitzar les estructures i relacions que hi ha a la KB, per identificar i emmagatzemar aquelles entitats que estan fortament relacionades generant una estructura de dades que finalment utilitzarem per a l'expansió de les consultes.

El mòdul Coordinador és qui s'encarrega de sincronitzar tots aquests processos per garantir un bon funcionament del sistema. Veurem més detalladament tots aquests mòduls durant el Capítol 8.

7.3 Comunicacions

Al tractar-se d'un sistema modular, la comunicació entre els diferents mòduls és un dels factors més crítics, ja que un mal funcionament de les comunicacions comportaria una mala sincronització entre els diferents processos, obtenint com a resultat un funcionament del sistema erroni.

Enrich Data utilitza dos tipus de comunicacions, via HTTP o mitjançant l'eina NSQ (5).

Com és habitual per realitzar les comunicacions amb una API, utilitzem el protocol HTTP per interaccionar amb els usuaris, amb mètodes GET, POST i DELETE.

La resta de comunicacions internes entre els diferents mòduls, les gestionem amb l'eina NSQ, una eina que permet la distribució de missatges en temps real, amb una gran estabilitat i escalabilitat.

Capítol 8

Mòduls

En aquest capítol, explicarem el sistema mòdul per mòdul, definint més detalladament les funcions que realitza cadascun, el seu disseny i la interacció amb les diferents bases de dades.

8.1 Mòdul Coordinador

Aquest mòdul és la peça central del sistema. Té dues funcionalitats principals, s'encarrega de gestionar les interaccions amb els webmasters i els administradors d'Enrich Data mitjançant una API, i de sincronitzar tots els processos que transcorren al sistema.

S'ha implementat amb el llenguatge de programació Go, un llenguatge de programació compilat amb tipat estàtic, utilitzant el framework Revel que ens facilita la implementació de l'API.

Primer veurem com aquest mòdul gestiona la sincronització de la resta de mòduls, i seguidament l'API que s'ha implementat perquè Enrich Data pugui interactuar amb els diferents actors.

8.1.1 Sincronització dels processos i gestió d'estats

Com ja havíem introduït, aquest mòdul és el responsable de garantir una correcta sincronització entre la resta dels mòduls. Mitjançant el pas de missatges indica al mòdul pertinent el procés a executar, i espera la seva resposta abans d'iniciar-ne un altre.

Al utilitzar el NSQ per a comunicar els mòduls, l'únic que hem de procurar és enviar els missatges pels canals correctes i així no equivocar-nos de destinatari, i identificar correctament les seves respostes. Es crea un canal per a cada mòdul, i cada mòdul es manté a l'espera de rebre missatges a través del seu canal. El mateix NSQ, s'encarrega de gestionar les cues de peticions i els possibles *timeout* que es puguin generar durant períodes d'estrès al sistema.

Quan una web es registra al sistema, li assignem un estat, que descriu el punt en el qual es troba el servei per a la web. Aquesta informació és útil tant pels clients que poden veure si el servei per a les seves webs ja està llest, com pels administradors del sistema que en el cas que produeixi un error o un procés es demori més del temps esperat puguin identificar ràpidament en quin mòdul es troba l'error.

Hi ha dos tipus d'esdeveniments que poden modificar l'estat associat al servei d'una web, les crides a l'API realitzades pels clients (API-Call:) i els esdeveniments interns del sistema, produïts com a conseqüència de l'inici o la finalització d'un procés (End-P:).

A la Figura 7 podem veure el diagrama d'estats.

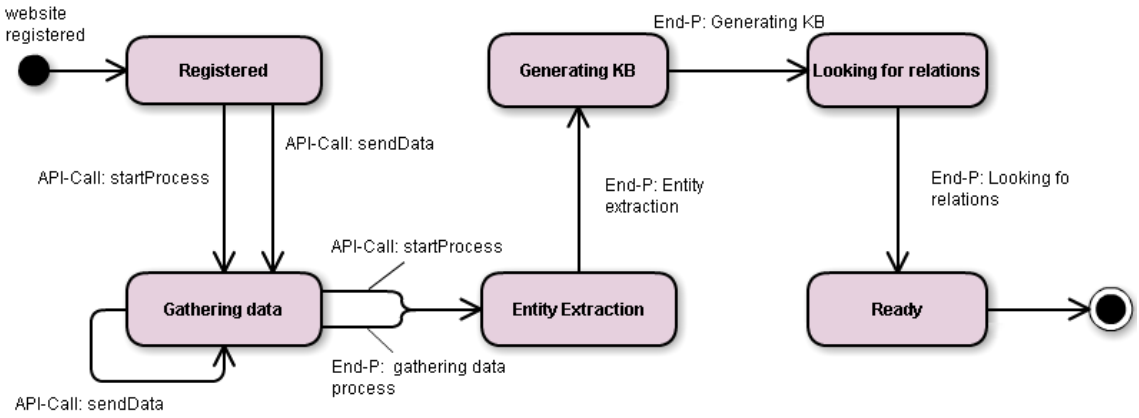


Figura 7 Diagrama d'estat del servei d'una web a Enrich Data.

Definició dels estats:

- "Registered": És l'estat inicial que li assignem a una plana web quan es registra al sistema
- "Gathering data": Indica que el sistema està executant el procés automàtic d'extracció i emmagatzemament del contingut de la web, o que el webmaster ha decidit enviar el contingut manualment i el sistema està a l'espera de rebre una petició per emmagatzemar més contingut o per iniciar el següent procés (identificació de les entitats rellevants de la web).
- "Entity extraction": Aquest estat indica que Enrich Data està executant el procés d'identificació de les entitats rellevants de la web.
- "Generating KB": Indica que s'està generant la KB de la web.
- "Looking for relations": Aquest estat senyala que el sistema ja ha generat la KB i que s'està executant el procés de cerca d'entitats fortament relacionades.

8.1.2 Api

L'API és la implementació de la interfície de comunicacions entre el sistema i els usuaris, i d'ella depèn que aquesta interacció sigui senzilla i fluida. Seguidament, es defineixen les diferents crides que formen l'API.

Mètode	Crida	Paràmetres	Resposta
POST	/register	url i contrasenya	id
Aquesta crida permet als clients registrar noves webs al sistema, indicant la url de la web que volen registrar i una contrasenya. Un cop feta la crida, la web queda registrada al sistema, i es retorna a l'usuari un identificador per a la web dins del sistema.			

Mètode	Crida	Paràmetres	Resposta
DELTE	/register	id i contrasenya	msg
Permet als clients eliminar les seves webs anteriorment registrades al sistema, indicant l'identificador de la web junt amb la contrasenya, o a l'administrador d'Enrich Data eliminar-ne qualsevol. Retorna al client un missatge indicant si s'ha processat bé la petició.			

Mètode	Crida	Paràmetres	Resposta
GET	/state	id i contrasenya	state
Mitjançant aquesta crida, els clients i l'administrador del sistema poden consultar l'estat en què es troba el servei per a una web, indicant l'identificador de la web junt amb la contrasenya.			

Mètode	Crida	Paràmetres	Resposta
GET	/queryExpansion	id, contrasenya i query	query, old[], new []
Aquesta crida permet als clients realitzar l'expansió de consultes, indicant l'identificador de la web, la contrasenya i la consulta. Es retorna a l'usuari la consulta original, junt amb dues *arrays*, una conté les entitats que hem extret de la consulta, i a l'altra els termes d'expansió resultants de fer l'expansió.			

Mètode	Crida	Paràmetres	Resposta
POST	/sendData	id, contrasenya i data	msg
Permet als clients enviar manualment el contingut de la web que volen que el sistema processi. A la crida cal indicar l'identificador de la web, junt amb la contrasenya i el contingut que es vol processar (data). Una vegada feta la crida el contingut s'emmagatzema a la base de dades i es retorna un missatge al client indicant si la petició s'ha processat correctament.			

Mètode	Crida	Paràmetres	Resposta
GET	/startProcess	id i contrasenya	msg
Amb aquesta crida els clients inicien el procés automàtic que prepararà l'expansió de consultes per a la web indicada. Si el client ha utilitzat la crida sendData per enviar manualment el contingut de la web, el sistema no realitzarà la fase d'emmagatzemament del contingut de la web, en el cas contrari començarà pel procés d'emmagatzematge. El client ha d'indicar l'identificador de la web i la contrasenya. Finalment es retorna un missatge al client indicant si la petició s'ha processat correctament.			

Mètode	Crida	Paràmetres	Resposta
GET	/webs	contrasenya	Webs[]
L'administrador del sistema pot consultar amb aquesta crida quines webs hi ha registrades al sistema i la informació emmagatzemada d'aquestes.			

Mètode	Crida	Paràmetres	Resposta
GET	/logs	id i contrasenya	Logs: []log
Aquesta crida permet a l'administrador del sistema consultar els logs que mostren estadístiques com el nombre de peticions processades d'una web indicant l'identificador de la web i la contrasenya.			

8.1.3 Diagrama de classes

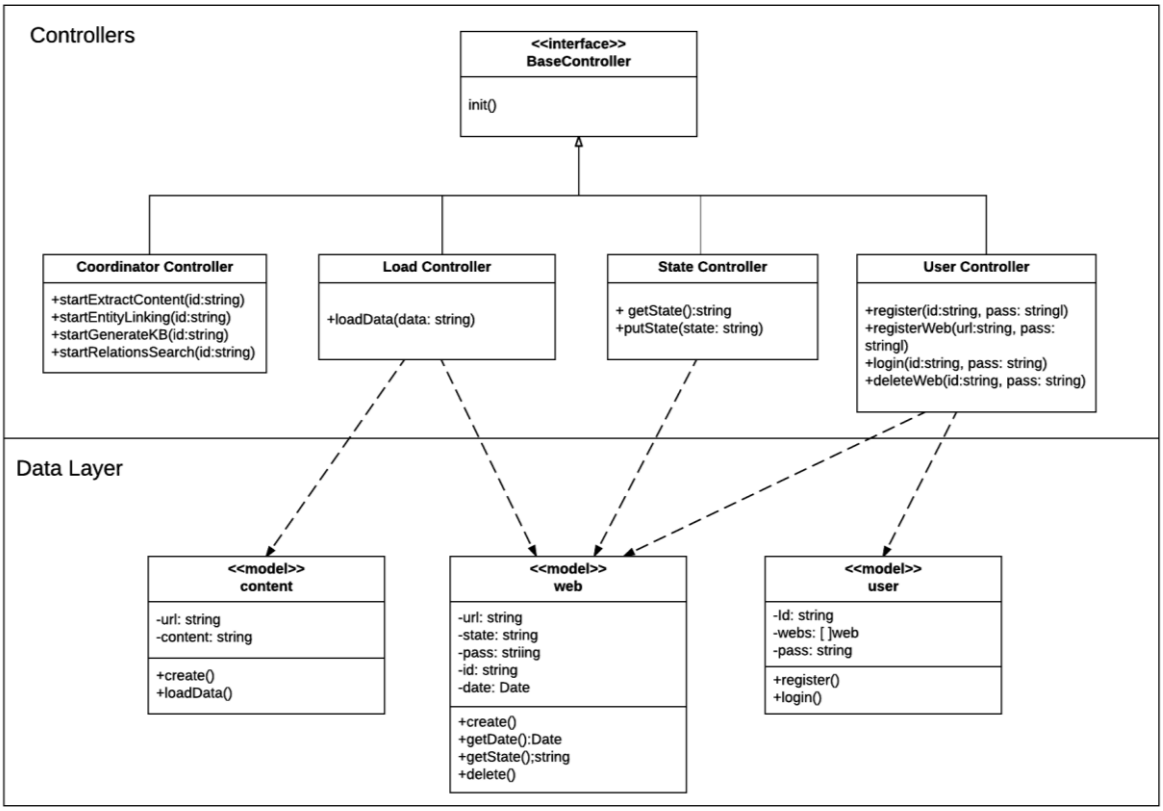


Figura 8 Diagrama de classes del mòdul Coordinador.

8.1.4 Descripció de les classes

Load Controller
Aquest controlador s'encarrega d'emmagatzemar a la base de dades el contingut de les webs que els webmasters envien manualment al utilitzar la crida /senData.

User Controller
S'encarrega de realitzar totes aquelles accions relacionades amb la gestió de les webs i usuaris registrats al sistema, com el registre de noves webs, o l'eliminació d'una. També s'encarrega de comprovar que les contrasenyes de les crides que rep el sistema siguin correctes.

State Controller
El State Controller gestiona les operacions relacionades amb les consultes i els canvis dels estats dels serveis de les webs registrades a Enrich Data.

Coordinador Controller
Aquest controlador s'encarrega de comunicar a la resta de mòduls d'Enrich Data quan han d'iniciar els seus processos.

8.1.5 Estructura de la DB

Aquest mòdul utilitza MongoDB per guardar la informació de les webs que es registren al sistema. Com que MongoDB és una base de dades orientada a documents, ens facilita molt el disseny de l'estructura. S'ha decidit generar una col·lecció a la qual anem afegint un document nou per a cada web nova que es registra al sistema, anomenada Webs. Podem veure aquest esquema a la Figura 9.

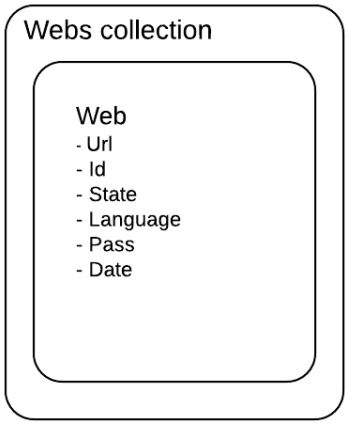


Figura 9 Esquema col·lecció Webs.

8.2 Mòdul Extractor de contingut

L'extractor de contingut s'encarrega d'emmagatzemar automàticament el contingut de les diferents pàgines que composen una web.

8.2.1 Funcionament

Aquest mòdul està constantment a l'espera de rebre un missatge via NSQ, que li indiqui sobre quina web ha d'iniciar el procés. Un cop rep aquest missatge amb l'identificador, accedeix a la base de dades per obtenir la url de la web i s'inicia un procés iteratiu que recorre cada un dels enllaços de la web emmagatzemant el seu contingut. La dificultat d'aquests procés rau en els següents factors.

- Cal garantir que els enllaços al que s'accedeixen són del domini de la web i no d'una altra, ja que una web pot contenir enllaços que apunten a altres webs.
- Cal gestionar els possibles errors que ens podem trobar al accedir als enllaços, atès que alguns servidors poden denegar el servei per un excés de peticions o simplement un enllaç pot estar corrupte.
- Cal controlar quin tipus de contingut conté un enllaç, ja que poden ser imatges, fitxers o documents, el contingut dels quals encara no analitzem. La capacitat de poder obtenir el contingut de documents com pdfs o doc és una funcionalitat que es preveu realitzar en el futur.

Aquest seria el pseudocodi del procés:

```
Inici
        //url conté la url de la web
        H<-hash
        q <- queue
        q.push(url)
        While q no està buida:
                u = q.pop()
                If H(u) == false:
                        H(u) = true
                        loadContent(u) //funció que emmagatzema el contingut d'una url
                        urls = getEnllaços(u) //funció que retorna els enllaços del
                                                contingut d'una url
                        q.push(urls)
Fi
```

Snippet 1 Pseudocodi de l'extracció del contingut d'una web

Un webmaster té dues opcions al utilitzar Enrich Data respecte com s'emmagatzema el contingut d'una web, pot escollir l'opció automàtica que hem definit anteriorment, o pot enviar el contingut manualment amb la crida a l'API sendData i saltar-se així el procés realitzat per aquest mòdul. També, com veurem en el Capítol 10, s'ha implementat un *plug-in* per a WordPress (6) que automatitza tot aquest pas i facilitar la càrrega del contingut a Enrich Data directament des de la base de dades, evitant així tot el procés de recórrer la web emmagatzemant els continguts.

8.2.2 Diagrama de classes

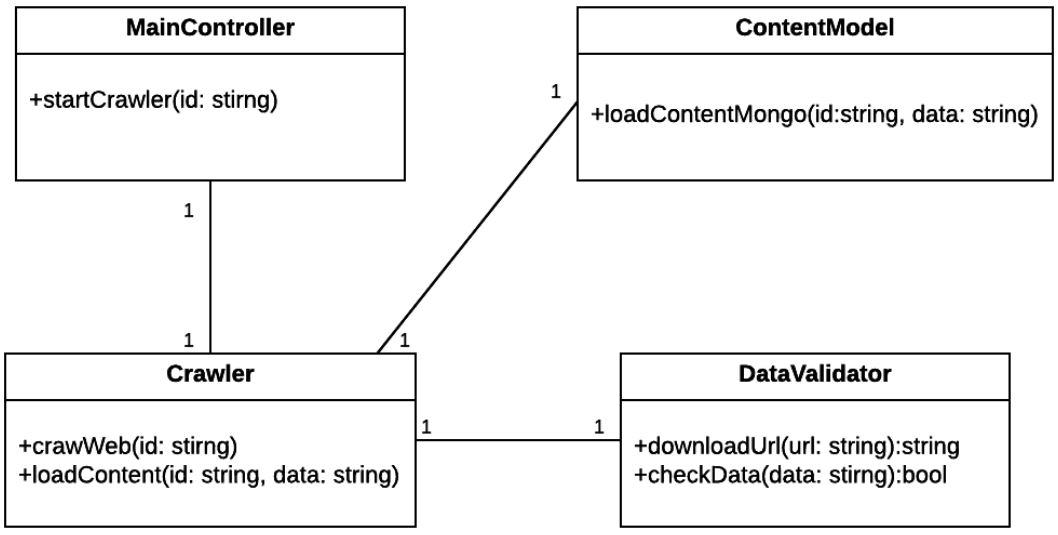


Figura 10 Diagrama de classes mòdul Extractor de contingut.

8.2.3 Descripció de les classes

MainController
És el controlador principal del mòdul, s'encarrega d'iniciar el procés d'emmagatzemament del contingut d'una web, creant una instancia de la classe Crawler.

Crawler
Aquesta classe és qui s'encarrega d'executar el pseudocodi definit en el anterior apartat, realitzant el procés emmagatzemament del contingut d'una web.

DataValidator
Aquesta classe donada una url té la funció de descarregar el seu contingut, gestionant els possibles errors que es puguin generar, i validant que el contingut descarregat és vàlid per emmagatzemar-lo a la base de dades.

ContentModel
Aquesta classe s'encarrega d'interactuar amb la base de dades, per emmagatzemar el contingut de les webs.

8.2.4 Estructura de la DB

L'Extractor de contingut també utilitza la col·lecció Webs de MongoDB especificada al mòdul Coordinador, però a més de la informació emmagatzemada a la col·lecció Webs durant el registre, ara també necessitem emmagatzemar el contingut de les webs.

MongoDB té un límit màxim de 16 megabytes per document, per satisfer aquesta restricció hem decidit generar una col·lecció per a cadascuna de les webs. Cadascuna d'aquestes col·leccions conté tants documents com enllaços conformen el lloc web. Amb aquest esquema, cada document emmagatzema únicament el contingut d'un enllaç satisfent així la restricció dels 16 megabytes per document. A la Figura 11 podem veure aquest esquema de la DB.

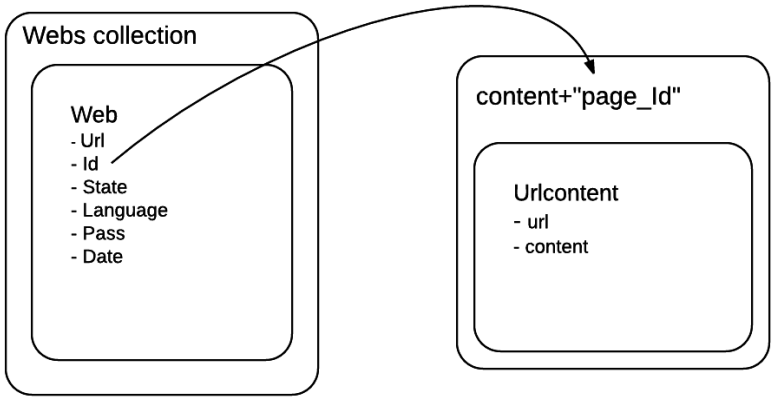


Figura 11 Esquema de les col·leccions "content".

8.3 Mòdul Extractor d'entitats

L'extractor d'entitats s'encarrega d'identificar les entitats més rellevants del contingut de les webs.

8.3.1 Funcionament

Per entendre aquest mòdul hem d'entendre a què ens referim quan parlem d'una entitat rellevant i com s'extrauen del contingut d'una web.

Definim entitats com aquelles paraules o conjunts de paraules que tenen sentit per si mateixes i són reconegudes per una ontologia, en aquest cas la Wikipedia. Quan es diu que una entitat està reconeguda per la Wikipedia, vol dir que hi ha un article a la Wikipedia que li fa referència. Per exemple, "java" seria una entitat perquè a la Wikipedia hi ha l'article "Java (programming language)" que li fa referència.

Cada article de la Wikipedia, va lligat a un identificador únic, així doncs quan seleccionem una entitat d'un text, aquesta va directament relacionada amb un article de la Wikipedia, que alhora té un identificador assignat, els quals més endavant utilitzarem per identificar-les.

Quan diem "les entitats més rellevants", ens referim a aquelles entitats més significatives pel text que s'està processant, que permetem distingir més clarament el contingut del text processat d'altres possibles textos.

Per fer aquesta extracció d'entitats utilitzem l'eina Dexter (7), un Framework Open Source per a Entity Linking. Mitjançant la seva API realitzem l'extracció d'entitats fent crides HTTP, amb les quals li enviem el text del qual volem extreure entitats, i Dexter ens retorna com a resposta el conjunt d'entitats que ha identificat junt amb els identificadors dels articles de la Wikipedia als qui fan referencia i els conjunts de paraules dels quals ha extret cada entitat, aquests conjunts de paraules els anomenarem "appear name".

Aquest seria un exemple real d'extracció d'entitats utilitzant l'eina Dexter on podem veure subratllades les entitats que han estat seleccionades:

Volkswagen was founded in 1937 to manufacture the car which would become known as the Beetle.

Com podem veure s'han identificat tres entitats, la primera "Volkswagen" que fa referencia al fabricant d'automòbils, "car" que fa referencia a cotxe, i Beetle, que fa referencia a un model de cotxe de Volkswagen.

Aleshores, per extreure les entitats d'una web, recorrem iterativament la col·lecció que emmagatzema el seu contingut, enviant al Dexter (1) seqüencialment els continguts de les url i emmagatzemant els resultats a la base de dades.

Aquest procés té dues limitacions, actualment només podem extreure entitats de texts en anglès, tot i que coneixem el mètode per poder adaptar Enrich Data a altres idiomes s'ha assignat aquesta tasca com a treball futur. La segona limitació consisteix en el fet que la

precisió de la identificació de les entitats en un text no és del 100%, a causa de la complexitat intrínseca del procés d'*entity linking*. Hem estudiat i configurat els diferents paràmetres del Dexter, obtenint una precisió del 96% (8).

8.3.2 Diagrama de classes

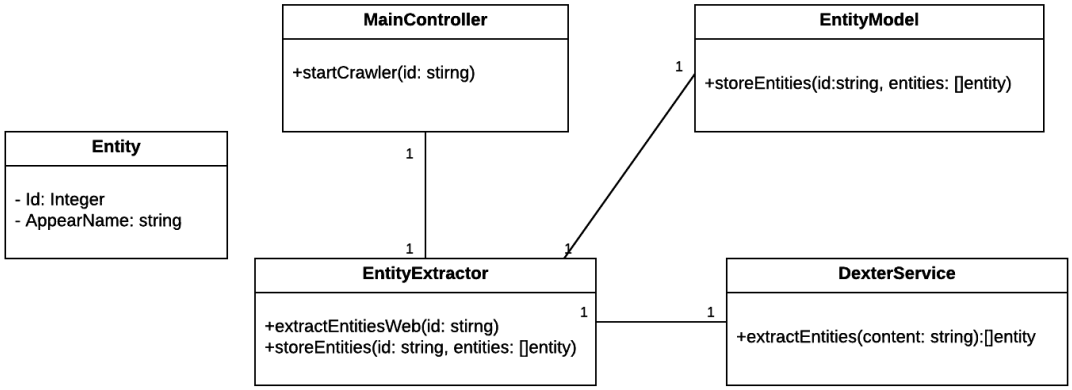


Figura 12 Diagrama de classes mòdul Extractor d'entitats.

8.3.3 Descripció de les classes

MainController
És el controlador principal del mòdul, s'encarrega d'iniciar el procés d'extracció de les entitats del contingut d'una web generant una instancia de la classe EntityExtarctor.

EntityExtractor
Aquesta classe s'encarrega de recórrer tot el contingut de la web indicada, extraient i emmagatzemant les entitats del seu contingut.

DexterService
Aquesta classe donat un text, utilitza l'eina Dexter per extreure-li les entitats.

EntityModel
EntityModel s'encarrega d'interactuar amb la base de dades, per emmagatzemar les entitats que s'han extret del contingut de les webs.

8.3.4 Estructura de la DB

Aquest mòdul utilitza les col·leccions content+"page_ID" descrites a l'anterior mòdul per accedir al contingut de la web que està processant.

Cal emmagatzemar les entitats extretes per Dexter, junt amb els identificadors dels articles als que fan referència. Seguint la mateixa metodologia que en l'apartat anterior, hem decidit generar una col·lecció nova per a cada web, on guardem un document per a cada enllaç de la web, emmagatzemant la seva url més els identificadors de les entitats que s'han extret del seu contingut junt amb els seus "appear name". A la Figura 13 podem veure aquest esquema de la DB.

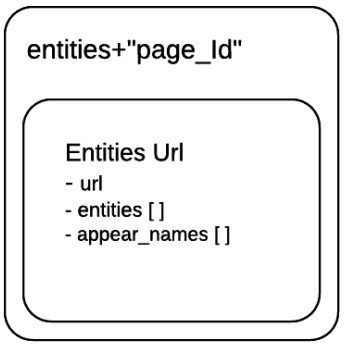


Figura 13 Esquema de les col·leccions "entities".

Amb aquesta estructura evitem també els problemes amb les mides màximes dels documents a MongoDB.

8.4 Mòdul Generador de KB

Aquest mòdul té com a objectiu seleccionar i emmagatzemar per a cada web registrada al sistema, tota aquella informació que més endavant pot ser rellevant per realitzar l'expansió de consultes. Aquest conjunt d'informació que emmagatzemem és el que anomenem KB, i és específica per a cadascuna de les webs registrades.

8.4.1 Generació del graf de la Wikipedia

Les KB es generen a partir del contingut de la Wikipedia. Per poder accedir als coneixements de la Wikipedia, l'hem carregat a Sparksee en forma de graf, per així poder navegar per la seva estructura. Per fer la carrega de la Wikipedia a Sparksee, en primer lloc hem processat una sèrie d'arxius[8] que la Wikipedia posa a la disposició de tothom que contenen tota la seva informació. Mitjançant Wikiparser (9), una eina que hem desenvolupat en el marc d'aquest projecte, obtenim un conjunt de fitxers en format CSV a partir dels arxius de la Wikipedia. El mòdul Generador graf Wikipedia, serà qui s'encarregui d'utilitzar aquests arxius CSV per generar el graf a Sparksee.

8.4.2 Estructura del graf de la Wikipedia

Seguidament veurem quina és l'estructura del graf de la Wikipedia que s'ha generat. La Wikipedia conté tres tipus d'elements diferents que es relacionen entre ells, els articles, les categories i els redirects

Quan fem una cerca a la Wikipedia i accedim a un contingut estem accedint a un article. Aquests articles poden contenir links que fan referència a altres articles. A la Figura 14

[8] https://dumps.wikimedia.org/

podem veure un exemple de tres articles de la Wikipedia.

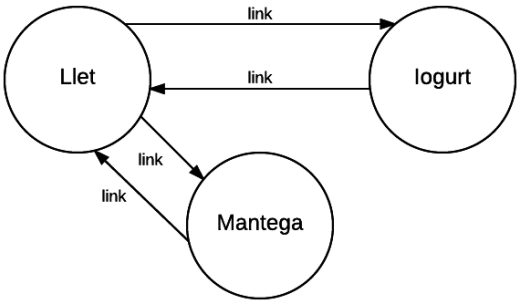


Figura 14 Estructura entre nodes Article al graf de la Wikipedia.

Les categories proposen una temàtica, i sota aquesta agrupen un conjunt d'articles. Un article pertany a com a mínim a una categoria, i les categories poden contenir subcategories i ser subcategories d'altres categories. A la Figura 15 podem veure dues categories relacionades entre elles (les categories són els nodes quadrats).

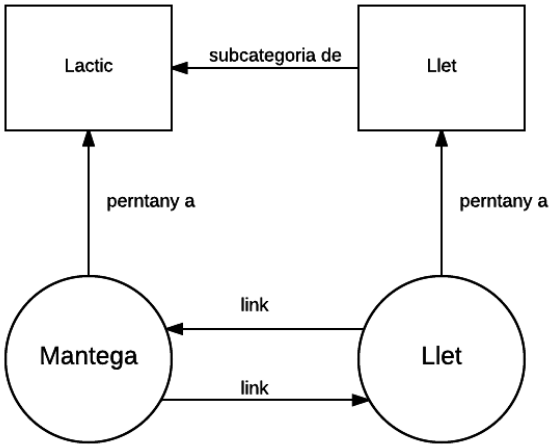


Figura 15 Estructura entre nodes Article i Categoria al graf de la Wikipedia.

Per últim, els redirects són un tipus especials d'articles, no tenen contingut i únicament apunten amb un link a un article. La Wikipedia utilitza els redirects per redireccionar possibles errors tipogràfics, o per exemple quan per referir-se a un mateix tema existeixen diferents noms els menys comuns s'emmagatzemen com a redirects del principal. Podem veure a la Figura 16 un exemple d'un redirect, que en aquest cas que redirigeix a l'entitat llet.

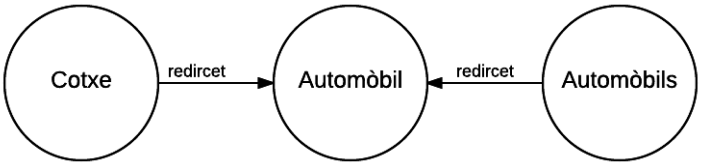


Figura 16 Estructura entre nodes Article i Redirect al graf de la Wikipedia.

8.4.3 Generació de la KB

Generar la KB d'una web es pot traduir a la pràctica com la selecció d'un subgraf dins del graf de la Wikipedia, obtingut com a resultat de seleccionar la informació del graf que s'utilitzarà per identificar les entitats fortament relacionades.

La dificultat d'aquest procés rau en decidir quins són els coneixements de la Wikipedia que formen part del domini de la web i quins no, delimitant així el subgraf dins de la Wikipedia.

Com s'ha vist anteriorment, cada entitat extreta del contingut d'una web va lligada a un article de la Wikipedia i per tant fa referència a un node del seu graf. Partint de l'anterior premissa, el primer pas del procés de generació d'una KB consistirà en seleccionar aquells articles dins del de la Wikipedia als que fan referència les entitats extretes de la web, i un cop s'han seleccionat, cal definir un algorisme que a partir d'ells delimiti el subgraf, decidint què seleccionem.

En la proposta actual, després d'estudiar les possibles opcions hem optat per a cada article seleccionat al graf de la Wikipedia a partir de les entitats que s'han extret del contingut d'una web també seleccionar:

- Tots els seus redirects.
- Tots els articles als quals fa referència amb un link junt amb els seus redirects.
- Totes les categories a les quals pertany l'article, juntament amb tots els articles i els seus redirects d'aquestes categories.

Així doncs s'ha implementat un algorisme iteratiu que recorre les entitats extretes del contingut de la web al graf de la Wikipedia, afegint aquestes a la KB de la web, junt amb els articles, categories i redirects que segueixen el patró definit anteriorment.

Cal veure que aquestes KBs al ser subgrafs del graf de la Wikipedia, tindran la seva mateixa estructura.

8.4.4 Diagrama de classes

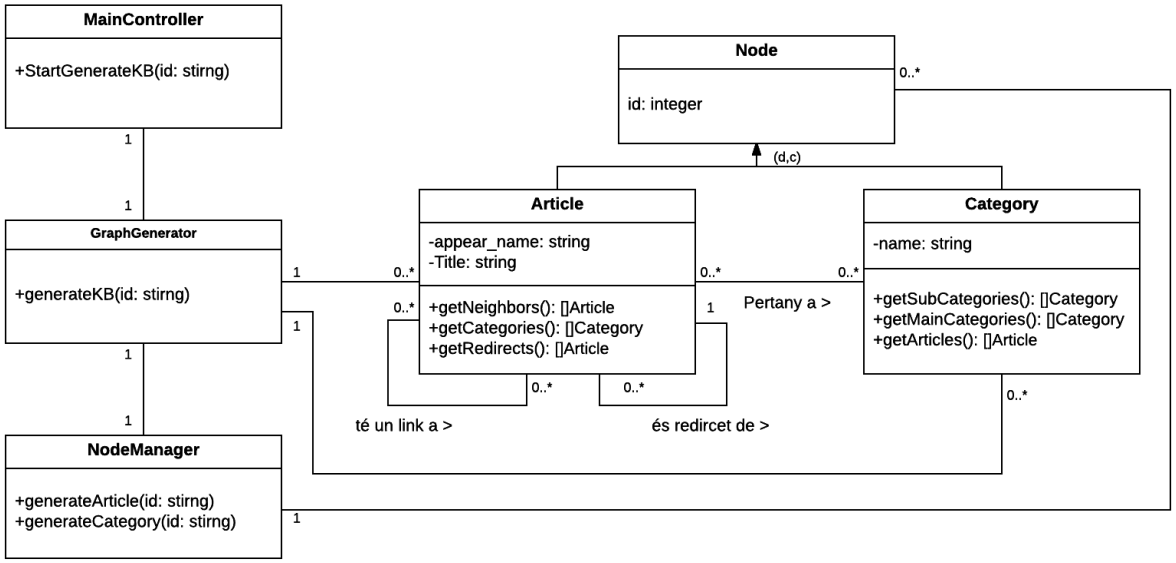


Figura 17 Diagrama de classes mòdul Generador de la KB.

8.4.5 Descripció de les classes

MainController
És el controlador principal del mòdul, s'encarrega d'iniciar el procés de generació de les KBs, inicialitzant i instanciant els principals objectes.

GraphGenerator
Aquesta classe és la que conté la càrrega algorítmica del mòdul, i s'encarrega de recórrer el graf de la Wikipedia per generar la nova KB.

NodeManager
El NodeManager s'encarrega de la instanciació de nodes del graf de la Wikipedia, i de la creació de nous nodes a les KBs.

Article
Representa en forma d'objecte els articles dels grafs, sobre els quals podem fer consultes per obtenir els articles amb els que estan relacionats, les categories a les quals pertanyen, els redirects que tenen, o a qui apunten com a redirects.

Category
Representa en forma d'objecte les categories dels grafs, sobre les que podem fer consultes per obtenir els articles i les subcategories que contenen i les categories a les que pertanyen.

8.4.6 Estructura de la DB

S'ha vist que aquest mòdul utilitza la base de dades orientada a grafs i quina és l'estructura de la KB, tot i així a la Figura 18 definim una estructura més genèrica del graf per tenir una visió més general de l'esquema utilitzat.

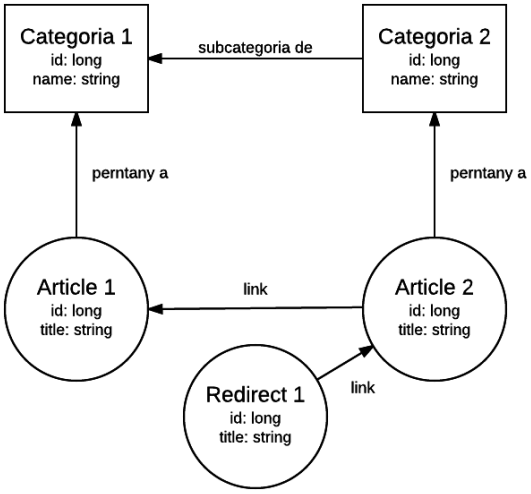


Figura 18 Estructura del graf de la Wikipedia.

8.5 Mòdul Cercador de relacions

Aquest mòdul s'encarrega d'analitzar les KBs amb l'objectiu d'identificar els parells d'articles que estan fortament relacionats, indexant aquests en una estructura de dades que s'utilitzarà durant les expansions de les consultes.

Per a identificar els articles que estan fortament relacionats basarem el nostre treball en (10) i (11), que formen part de la recerca a la qual aquest TFG ha donat suport. En el primer dels articles els autors estudien, a partir d'un ground truth basat en un benchmark d'information retrieval, les relacions entre els articles que permetien obtenir els millors resultats utilitzant els seus títols com a termes d'expansió. D'acord a l'anàlisi, els articles que responien millor a les consultes del benchmark estaven relacionats entre si a partir de k-cicles, definits d'acord als autors com a una cadena tancada entre articles i categories amb com a mínim k arestes entre si. A més, els resultats de l'anàlisi revelaven que aquests cicles tenien unes característiques definides que resumim a continuació:

- Els cicles de longitud 2 no són fiables.
- Es creu que els cicles de longitud 3, 4 i 5 poden identificar els articles que estan fortament relacionats.
- Al voltant d'un terç dels nodes que formen els cicles han de ser categories. S'espera que aquesta proporció millori per cicles amb longitud major a cinc.

- Els termes d'expansió obtinguts a través d'articles compresos en cicles densos obtenen millors resultats.

En el segon article, els autors definien un conjunt de motius estructurals que complien les característiques abans definides. Aquests motius són la materialització de les característiques en estructures pròpies dels grafs. En concret els autors defineixen dos tipus de motius dibuixats a la Figura 19.

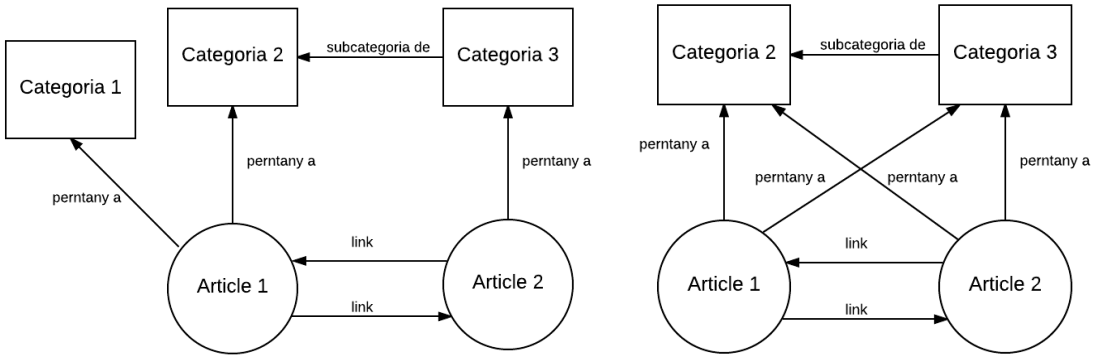


Figura 19 Motius d'expansió de l'article (motiu triangle a la dreta, i quadrat a l'esquerra).

El primer dels motius anomenat triangle consisteix en dos articles de la Wikipedia, els quals s'apunten recíprocament amb un link, on un d'ells pertany a totes les categories de l'altre. El segon dels motius anomenat quadrat consisteix també en dos articles que es fan referència recíprocament amb un link, però en aquest cas és suficient amb què dues de les seves categories estiguin relacionades, és a dir, que una sigui subcategoria de l'altre. Observem doncs que aquests motius compleixen les característiques prèviament descrites, i a més, d'acord amb els autors permeten millorar els resultats de la cerca fins a un 150%.

Notis que en el context de l'actual projecte no té sentit fer servir exactament les mateixes estructures. Mentre que els articles prèviament esmentats es focalitzen en millorar les consultes en un sistema genèric basat en la Wikipedia, nosaltres pretenem donar una solució personalitzada a cadascuna de les webs. Per això, en comptes de cercar estructures a la Wikipedia, les cerquem a les KBs de les webs. Fer un primer filtratge, elimina els articles no relacionats amb el domini de la web, això ens permet ser una mica més laxes en la definició de les estructures que utilitzarem. En concret, els motius que nosaltres fem servir per identificar articles fortament relacionats són el mateix motiu quadrat, i una versió del motiu triangle menys estricta, il·lustrada a la Figura 20. A l'estructura triangle que nosaltres utilitzarem, els articles també es fan referència recíprocament amb un link però és suficient que comparteixin una sola categoria, i no totes.

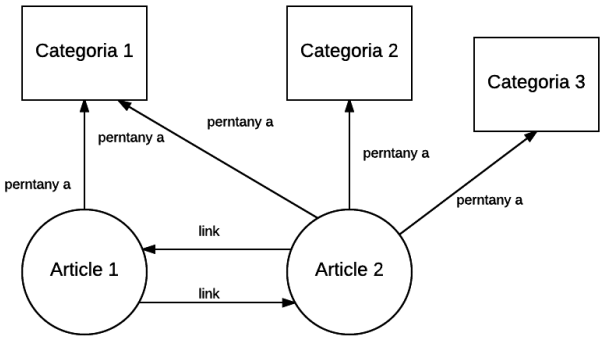


Figura 20 Motiu triangle Enrich Data.

Actualment el sistema sempre utilitza les mateixes estructures, tanmateix, en un futur volem treballar en millorar aquesta part perquè el propi sistema decideixi utilitzar unes estructures o unes altres a partir de les característiques de cada KB.

8.5.1 Funcionament

Aquest mòdul es manté a l'espera de rebre missatges del Coordinador que l'indiquin per a quina web iniciar el procés.

Aquest procés recorre totes les entitats de la web que hi ha a la seva KB, i per cada una d'elles cerca els articles amb els quals comparteixen un dels motius que s'han definit abans, i per tant estan fortament relacionats.

Aquest procés es realitzà amb un algorisme que de forma iterativa va recorrent cadascuna de les entitats de la web a la KB comprovant per cada un dels seus articles veïns si encaixen en alguns dels motius, i en el cas afirmatiu guardar aquest dos articles com una relació a la base de dades.

8.5.2 Diagrama de classes

Aquest mòdul està basat en la implementació del mòdul Generador de la KB, ja que tracten grafs amb la mateixa estructura. La diferència entre els mòduls a nivell d'implementació rau en què aquest nòdul té la classe *RelationSearcher* i no la *GraphGenerator*. Aquesta nova classe és qui s'encarrega de recórrer les KBs, cercant i emmagatzemant les entitats fortament relacionades.

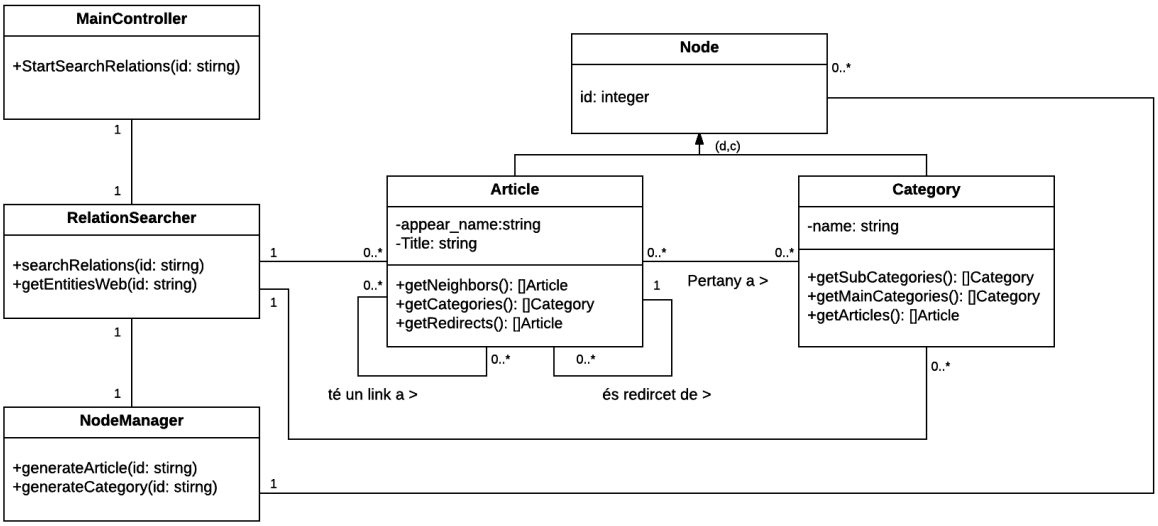


Figura 21 Diagrama de classes de mòdul Cercador de relacions.

8.5.3 Estructura de la DB

Aquest mòdul emmagatzema les relacions entre articles que estan fortament relacionats utilitzant MongoDB com a base de dades. S'ha decidit generar una col·lecció nova per a cada web, en les que cada document de les col·leccions emmagatzemarà les relacions que té una entitat, seguint l'estructura de la Figura 22. D'aquesta forma es genera a cada col·lecció una estructura que anomenarem "Entity relations table" *(ERT)*, en el que donada una entitat podem trobar amb quins articles està fortament relacionada.

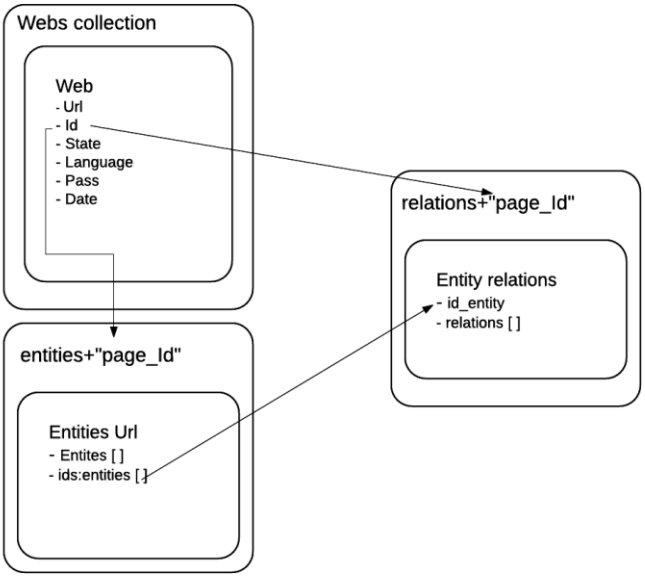


Figura 22 Esquema de les col·leccions "relations" de la DB.

Partint del que hem vist anteriorment, quan cerquem una relació, sempre partim d'una entitat que s'ha extret de la web. Per tant les parelles d'articles fortament relacionats que cerquem sempre contenen un primer article (e_1) que apareix a la web, i un segon (e_2) que

pot aparèixer o no. Quan emmagatzemem aquestes relacions al *ERT*, sempre guardem e_1 com a relació de e_2, guardant el *appear name* i no l'identificador de l'entitat, per així facilitar l'expansió de les consultes que veurem en el següent mòdul.

8.6 Modul Query expander

El Query expander s'encarrega de realitzar el procés d'expansió de consultes sobre les webs registrades al servei.

Aquest procés només es pot realitzar una vegada tots els mòduls anteriorment descrits hagin finalitzat les seves tasques, ja que necessita l'estructura de dades que emmagatzema les relacions entre les entitats fortament relacionades de la KB per poder realitzar les expansions.

Per a fer l'expansió de la consulta, el primer pas consisteix en identificar les entitats de la consulta, i una vegada les hem extret, expandir la consulta afegint com a nous termes aquelles entitats que estan fortament relacionades amb elles.

A causa del gran flux de peticions que pot rebre aquest mòdul (les peticions d'expansió de les cerques de les webs), s'ha decidit separar aquest de la resta, per guanyar estabilitat. Un gran flux de peticions entrants podria provocar un gran estrès al sistema, separant aquest mòdul reduïm les conseqüències de possibles problemes d'estabilitat causats per aquest estrès. Pensant en aquests problemes el mòdul Query expander s'ha dissenyat i desenvolupat per poder escalar horitzontalment i així poder donar servei a un gran flux de peticions.

8.6.1 Funcionament

Aquest mòdul es manté a l'espera de rebre peticions d'expansió de consultes via HTTP. Per cada petició rep la consulta escrita per l'usuari, i l'identificador de la plana web sobre la qual s'ha realitzat la consulta.

El primer pas consisteix en extreure les entitats de la consulta utilitzant Dexter. De la mateixa forma que s'ha vist a l'apartat del mòdul Extractor d'entitats, mitjançant una crida HTTP envien el contingut de la consulta a Dexter, que ens retorna les entitats que conté, junt amb els seus identificadors que fan referència a articles de la Wikipedia.

Un cop tenim el conjunt d'entitats que s'han extret de la consulta, accedim a la col·lecció que emmagatzema les relacions de la web sobre la qual es fa la consulta, i cerquem per a cada entitat les entitats amb les quals té una relació forta. Així doncs, tenim les entitats que s'han extret de la consulta junt amb els *appear name* de les entitats amb les que estan fortament relacionades que anomenem termes d'expansió, aquest conjunt és el resultat de l'expansió de la consulta. A la Figura 23 podem veure un diagrama del procés d'expansió.

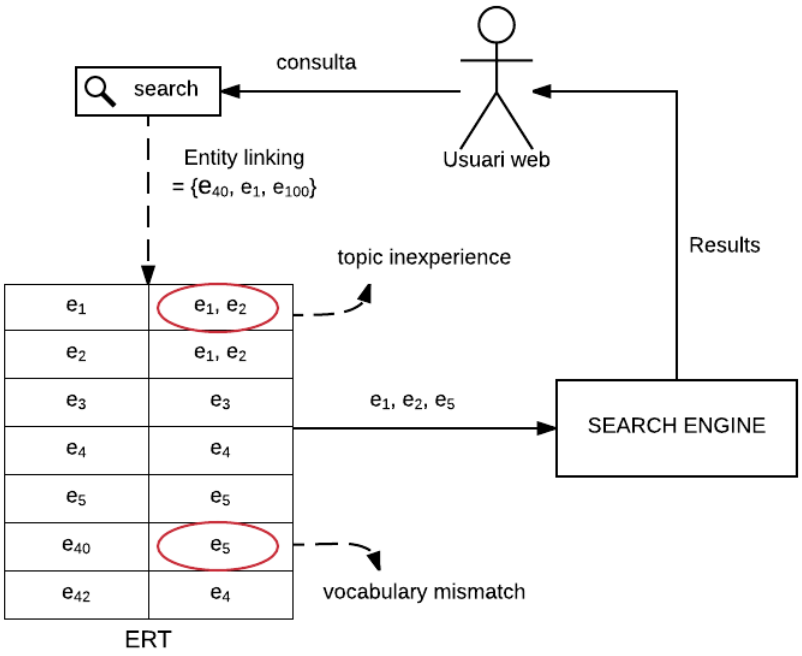


Figura 23 Procés d'expansió.

A l'accedir al *ERT* amb cada una de les entitats extretes de la consulta que poden aparèixer o no a la web, estem trobant aquelles entitats que sí que apareixen a la web amb les que estan fortament relacionades. D'aquesta forma quan un usuari fa una cerca que no retornaria cap resultat, com a la Figura 23 on un usuari a fet una consulta que conté l'entitat e_{40} que no apareix a la web, mitjançant l'expansió de la consulta trobaríem els *appear name* de les entitats fortament relacionades amb la consulta que sí apareixen com e_5, amb els quals sí que s'obtindrien resultats al fer la cerca. Aquest anterior cas seria una clara referència al primer escenari, *vocabulary mismatch*, definit durant la formulació del problema d'aquest projecte.

A la Figura 23 també podem veure una representació del segon escenari definit a la formulació del problema, *topic inexperience*. En aquest segon cas podem veure com l'usuari a causa de la falta de familiarització amb els temes de la cerca fa una consulta que conté una sola entitat que apareix a la web (e_1), i gràcies a l'expansió de la consulta també trobarà altres entitats fortament relacionades amb la seva consulta com (e_2).

Al ser una petició HTTP, la resposta també es retornada per aquesta via, en format JSON:

```
{
        query: string //conté la consulta escripta per l'usuari
        oldEntities: [] // conté una array amb les entitats extretes de la consulta
        newEntities: [] // conté una array amb els termes d'expansió seleccionats
}
```

S'ha decidit retornar els diferents components de les respostes per separats, donant flexibilitat als webmaster que poden gestionar-les com millor els hi convingui, i per exemple, si a l'expandir una consulta s'han trobat molts termes d'expansió, el webmaster podrà escollir-ne només una part, o seleccionar el text de la consulta original si no s'ha trobat cap.

8.6.2 Diagrama de classes

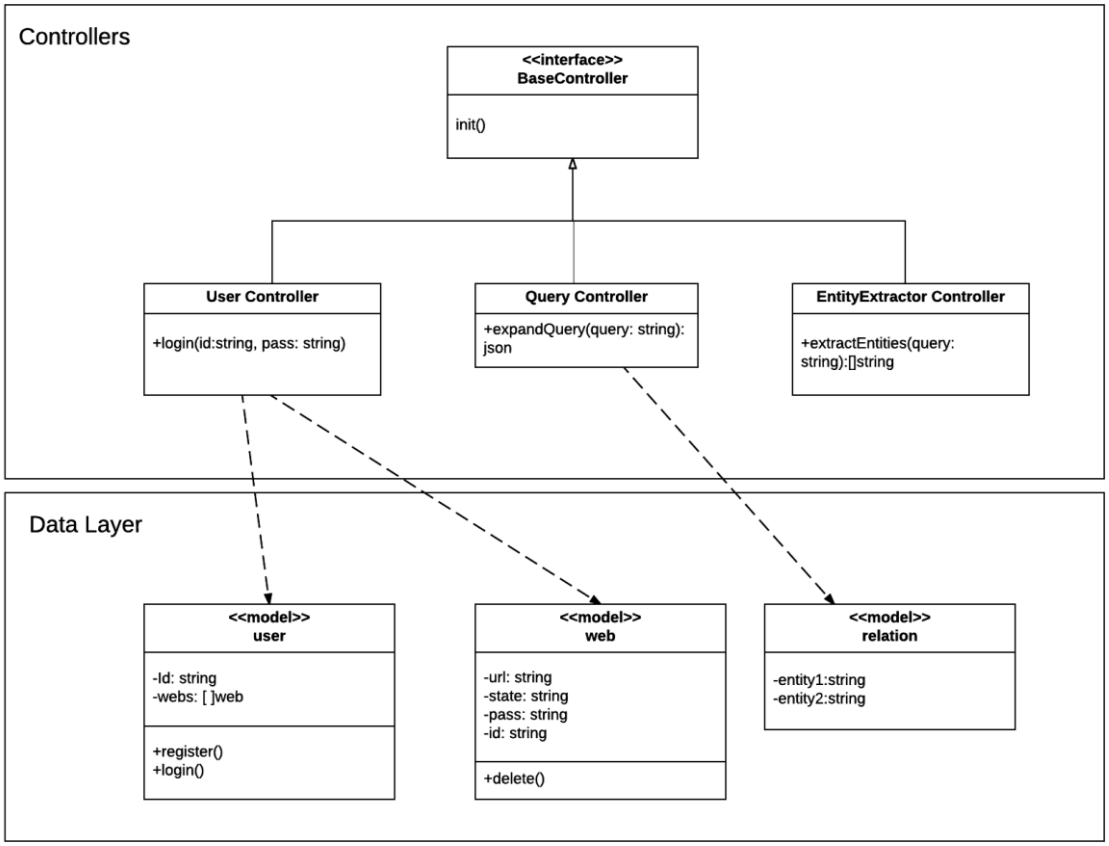


Figura 24 Diagrama de classes de mòdul Query Expander.

8.6.3 Descripció de les classes

Query Controller
Aquest controlador s'encarrega de gestionar les peticions d'expansió de consultes.

User Controller
S'encarrega de comprovar que la contrasenya de les crides que rep el sistema siguin correctes.

EntityExtractor Controller
Aquesta classe s'ocupa d'extreure les entitats de les consultes de les peticions d'expansió utilitzant Dexter.

8.6.4 Escalat del sistema

Com ja s'ha introduït anteriorment el mòdul Query expander pot escalar horitzontalment. Seguidament es presentarà com es realitzaria l'escalat, les diferents variants que hi ha, i quin benefici té cadascuna.

El procés d'expansió de consultes, només utilitza el mòdul Query Expander i les col·leccions de MongoDB on s'emmagatzemen les *ERT*, per aquest motiu a l'escalar el sistema només haurem de tindre en compte aquests dos elements.

Ens trobem amb dues variants diferents per escalar el sistema, segons la distribució de les *ERTs* als servidors. A la primera variant cada servidor només conté les *ERTs* d'unes webs en concret, i no les de tot el sistema. Com a benefici trobem:

- Les actualitzacions de *ERTs* són eficients, ja que no s'han d'actualitzar tots els servidors.
- Existeix la possibilitat de decidir més específicament las distribució de les peticions de les webs entre els diferents servidors.
- El consum d'espai al disc dels servidors és reduït.

Com a inconvenient trobem:

- Una mala distribució pot comportar col·lapsar alguns servidors més que altres i per tant donar un servei poc homogeni als clients.

La segona variant en canvi, guarda totes les *ERTs* en tots els servidors, així que tots els servidors poden donar servei a totes les webs registrades al sistema. Com a benefici trobem:

- El sistema distribueix la càrrega de forma automàtica donant la mateixa qualitat de servei a totes les webs

Com a inconvenient trobem:

- Les actualitzacions de *ERTs* són més complexes, ja que s'han d'actualitzar a tots els servidors.
- Quan més webs es registren al sistema, menys espai disponible hi haurà als servidors, fins al punt de no poder donar servei a més webs per falta d'espai al disc.

Veient els beneficis i inconvenients de les dues variants anteriors, s'ha dissenyat una tercera variant fusionant les dues anteriors. En aquesta tercera variant les webs es distribueixen en clústers de servidors que segueixen la distribució de la segona variant, es formem grup de servidors que comparteixen les mateixes webs. Amb aquesta variant tindríem els beneficis de la segona variant, però estalviant-nos el problema de l'espai disponible.

A la Figura 25 podem veure l'estructura d'un sistema amb clústers i dos servidors per clúster.

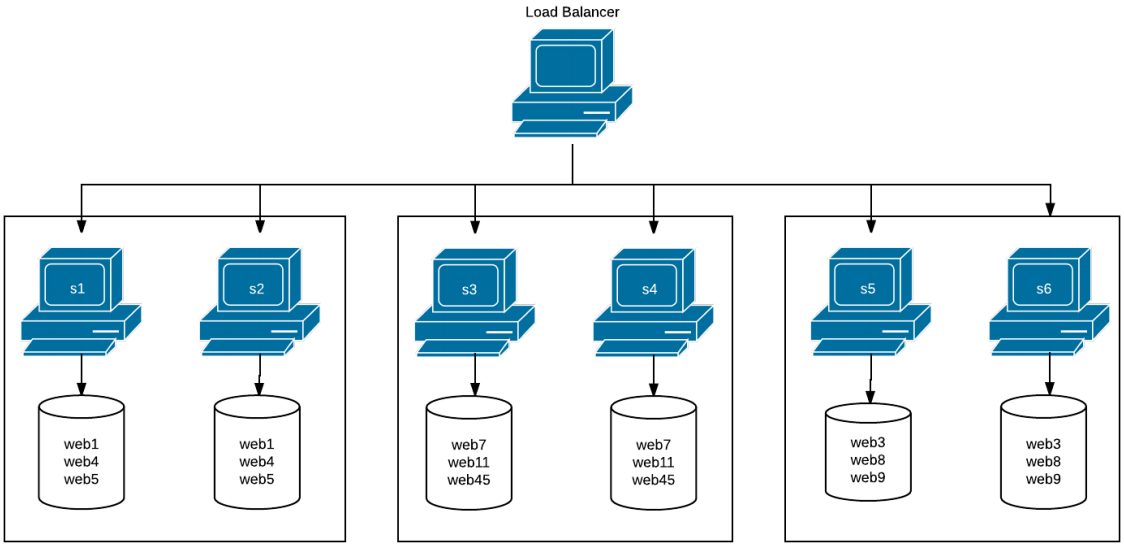


Figura 25 Estructura del sistema d'expansió amb tres clústers i dos servidors per clúster.

Com podem veure a la Figura 25, per distribuir la càrrega entre els diferents servidors fa falta un distribuïdor de càrrega, possiblement utilitzaríem un servidor amb Nginx (un tipus de servidor *proxy*) per a realitzar aquest paper.

Capítol 9

Sostenibilitat i compromís social

Matriu de sostenibilitat:

Econòmica	Social	Ambiental
8	8	9

9.1 Econòmica

L'objectiu d'aquest projecte, des del punt de vista comercial, consisteix en produir ingressos a partir de la comercialització del servei.

El preu del producte encara no està fixat, ja que cal fer un estudi sobre el mercat per veure a quins sectors d'aquest volem enfocar el producte, i així definir uns preus. Per una altra banda també s'ha de pensar sobre com es publicitarà el producte, amb quins medis i quins costos tindran. Considero que aquesta part comercial no entra dintre de l'abast aquest projecte.

En tot cas, donats els objectius del producte podem pressuposar que el preu del producte no serà molt elevat, per així ser més atractiu, i també es tractarà d'un sistema amb un cost de manteniment mínim, ja que escalar el sistema horitzontalment no suposa un cost alt, i reparar el sistema o restaurar-lo sobre un back-up seria molt senzill.

Pels anteriors motius considero que té una valoració de 8 sobre viabilitat econòmica.

9.2 Social

El servei que s'està desenvolupant té dos principals actors que en surten beneficiats.

Per una banda, tenim als webmasters, que obtindrien una millor incidència de la informació que proveeixen les seves webs, el que podria comportar uns majors ingressos en vendes, si es tracta d'una tenda online, o millors ingressos en publicitat si es tracta d'un blog al tenir més visites. Altrament, tenim els usuaris que visiten les webs, els quals es veurien beneficiats directament a causa de la millora sobre l'experiència d'ús al visitar les webs.

Valoraríem l'impacte social amb un 8.

9.3 Ambiental

Els únics recursos energètics que es consumiran durant el desenvolupament de projecte seran el consum elèctric del portàtil i dels servidors. Ja que el consum del portàtil és molt baix, i un dels servidors és propietat del departament del grup de recerca DAMA-UPC, i aquest es utilitzat paral·lelament per altres projectes, podem assumir que el cost energètic és mínim.

Pensant en el producte una vegada comercialitzat, serà un producte allotjat al cloud, la gestió i despesa del hardware seria gestionada directament pel grup de recerca DAMA-UPC, tenint cura de reaprofitar els recursos i conservar i reciclar màquines en desús o trencades.

Es considera que la sostenibilitat ambiental es podria valorar amb un 9.

Capítol 10

Qeast in action

Enrich Data s'ha desenvolupat al grup de recerca DAMA-UPC, amb el propòsit de donar suport a la recerca d'en Joan Guisado i dur a terme un projecte europeu finançat per la iniciativa Tetracom[9]. A banda d'aquest propòsit també existeix un interès comercial. Actualment, Enrich Data és totalment funcional, i es troba en una fase beta gratuïta adoptant com a nom comercial Qeast.

10.1 Web comercial

S'ha dissenyat i implementat una web comercial, on s'explica què és Qeast i com utilitzar-lo, "www.qeastsearch.com". Dins la web podem trobar la documentació de l'API, junt amb un formulari de registre per a nous clients i un *log in* per clients ja registrats (encara en desenvolupament). A la Figura 26 podem veure una imatge de la web.

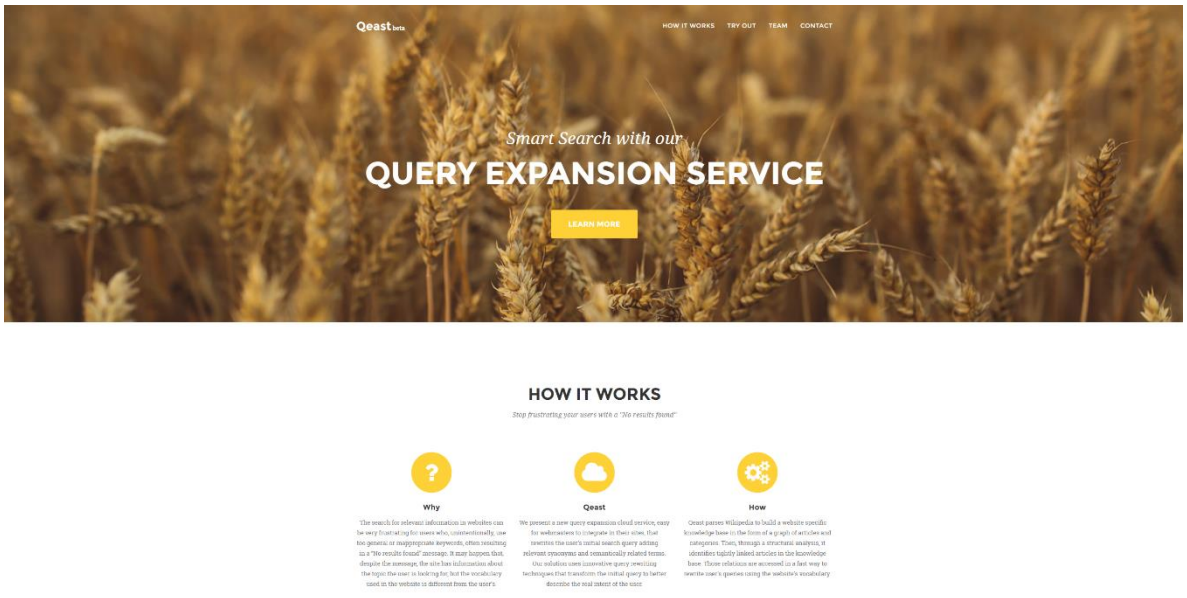


Figura 26 Web comercial de Qeast.

[9] http://www.tetracom.eu/

10.2 Demo

S'ha implementat una demo perquè potencials clients puguin fer consultes sobre dues webs diferents i comparar els resultats obtinguts al fer la consulta normal a les diferents webs i al fer-la utilitzant Qeast, podent comprovar de forma tangible els beneficis d'utilitzar Qeast. En la Figura 27 podem veure una imatge de la demo. S'han integrat dues webs sobre les quals realitzar les consultes:

- http://sparsity-technologies.com/blog/: la web comercial d'una empresa tecnològica especialitzada en grafs, creadors de la base de dades Sparksee, i estretament vinculats amb el grup de recerca DAMA-UPC.
- http://iamafoodblog.com/: Un blog de receptes culinàries.

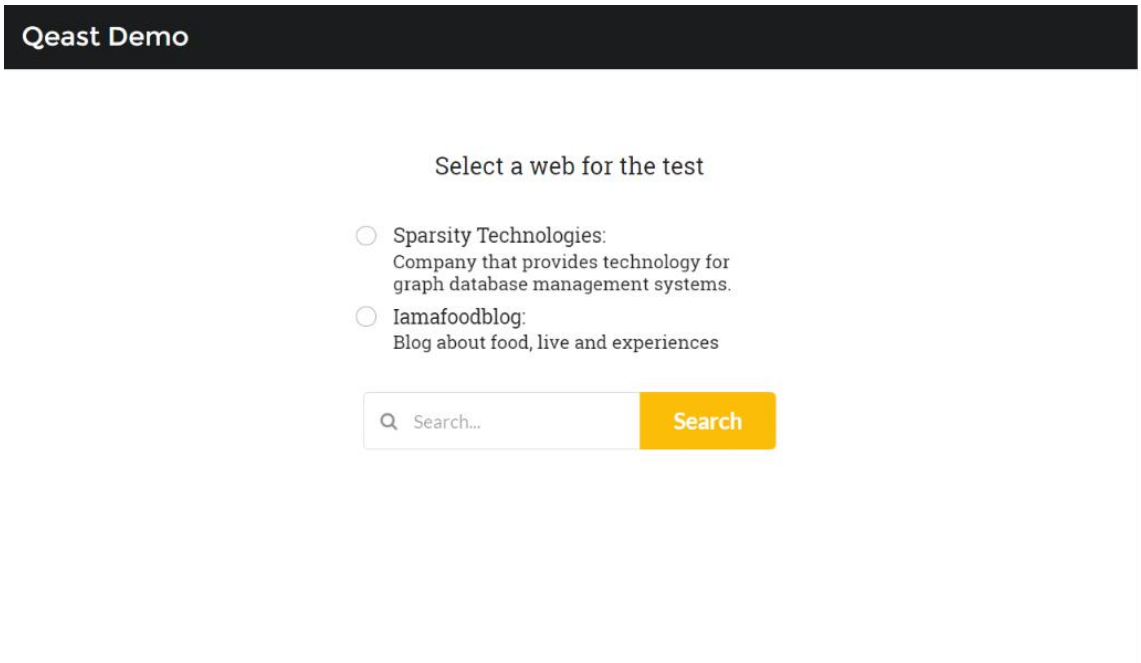


Figura 27 Web de la demo de Qeast.

Els dos exemples que formen la demo, s'han generat mitjançant el procés que efectuarien els futurs clients de Qeast. S'han registrat les webs al sistema seguint tot el procés descrit durant aquesta memòria. L'única diferencia respecte a un cas real, rau en què no hem modificat les webs perquè integrin Qeast. Així doncs, realitzem una simulació, cercant a les webs els termes d'expansió retornats per la petició, i mostrant seguidament tots els resultats junts. Aquesta cerca manual consisteix en cercar inicialment un per un els termes d'expansió a la web, i per finalitzar tots els termes junts en una sola cerca.

A la Figura 28 podem veure el resultat de buscar a la demo "sausage" sobre la web www.iamafoodblog.com, un blog sobre receptes de cuina.

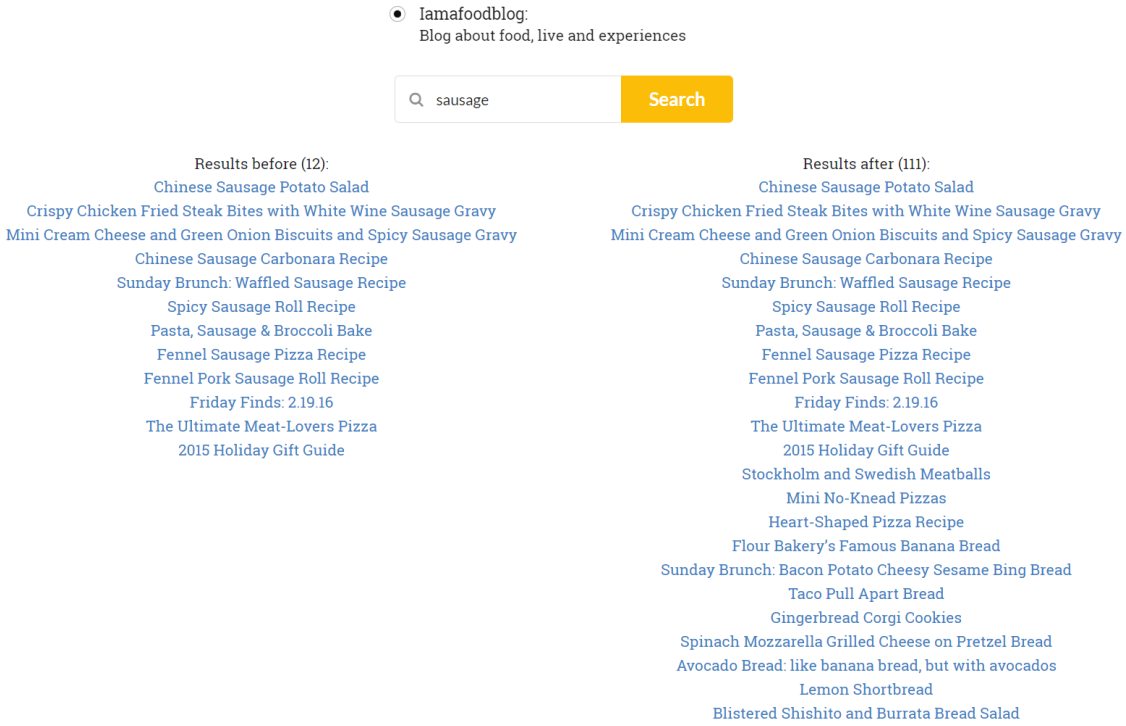


Figura 28 Resultat de la cerca "sausage" a la demo.

Podem observar que sense utilitzar Qeast s'han trobat 12 resultats en els quals únicament apareix "sausage", i en canvi utilitzant-lo 111, trobant entrades on apareixen entitats fortament relacionades com "hot dog", això és degut als termes d'expansió que s'han trobat al fer la petició a Qeast per expandir la consulta. A la Taula 1 es poden veure els resultats de les expansions de tres consultes.

Consulta	Termes d'expansió
sausage	merguez, sauerkraut, salami, pepperoni, bread, stuffing, pork, fry up, boudin, chinese sausage, italian sausage, pate, portuguese sausage, sausage rolls, breakfast sausage, hot dogs, andouille, chorizo, bacon, barbecue, smoked
milk	goat, evaporated milk, butter, ghee, dairy, buttermilk, condensed milk, dairy products, milk, sheep milk, cheese, yogurt, dairy allergies, curds
fuet	pork, sausage

Taula 1 Exemples d'expansió de consultes

A la segona consulta s'ha cercat "milk". Cal recordar que tots els termes d'expansió són entitats que apareixen a la web i per tant al fer l'expansió d'aquesta consulta, la cerca no només retornaria entrades que continguessin "milk", sinó que també retornaria altres continguts fortament relacionats en els que apareguessin els termes d'expansió trobats com "dairy" o "cheese". Com podem veure els termes d'expansió retornats en aquesta consulta estan relacionats amb la consulta realitzada, i podrien millorar i ampliar els seus resultats.

En el tercer cas, s'ha realitzat la cerca "fuet", al ser una web amb el contingut únicament en anglès no trobaríem cap resultat realitzant aquesta consulta. Al fer l'expansió s'han trobat dues entitats com a termes d'expansió, "pork" i "sausage". Utilitzant aquests termes d'expansió, seguiríem sense trobar entrades en les quals aparegui "fuet" però si trobaríem entrades al blog on apareixen els termes d'expansió anteriors que estan fortament relacionats amb la consulta. Això és gràcies al fet que el mòdul Generador de la KB, quan estava generant la KB pel blog culinari, ha afegit l'entitat "fuet" a la KB, que finalment ha resultat estar fortament relacionada amb "pork" i "sausage", i al fer l'expansió de "fuet" apareixen com a termes d'expansió.

10.3 Proves d'estrès

S'ha realitzat un estudi per determinar quin és el flux de peticions d'expansió que Qeast és capaç de servir, per poder analitzar quins són els recursos necessaris per a donar servei a un cert nombre de clients, per així poder fer un pla sobre com escalar el sistema quan el nombre de clients comenci a créixer.

S'ha analitzat el flux de les peticions d'expansió, perquè són les que poden arribar a tenir el cabal més gran podent generar un estrès al servidor que pot convertint-se en un punt crític del sistema.

Hem realitzat les proves sobre el servidor actual en el que es troba instal·lat Qeast que té el següent hardware:

Processador: Core i7-920, 4 cores a 2.66 ghz

Disc dur: 2 TB HD

Memòria ram: 16 GB

Amplada de xarxa: 100 Mbps

S'ha utilitzat Locust 10 com a eina per fer les proves d'estrès, un framework de python open source per fer proves de càrrega.

Els tests consisteixen en la simulació d'un *worst-case scenario* on un nombre determinat d'usuaris envien simultàniament peticions d'expansió de consultes a Qeast sobre qualsevol de les dues webs disponibles a la demo. Quan un usuari rep la resposta de la

[10] http://locust.io/

seva petició, en realitza un altre immediatament. Així, el que reproduïm són cues de peticions de mida fixa al servidor, és a dir, quan fem el test amb 10 usuaris estem reproduint una cua de 10 peticions simultànies al servidor, ja que quan una petició és atesa els usuaris envien una nova.

Aquesta simulació per tant, no pretén simular el comportament d'un conjunt d'usuaris reals, ja que no estem reproduint una conducta típica. El que busquem en aquestes proves és veure quin flux màxim de peticions que és capaç de servir Qeast amb un temps de resposta mig acceptable. Hem determinat que un temps de resposta acceptable ha de trobar-se per sota del 500ms, ja que en el cas contrari estaríem perjudicant l'experiència d'ús dels usuaris al alentir massa les cerques a les webs.

Test amb 10 usuaris: Temps mitjà = 437ms, peticions per segon: 24,2

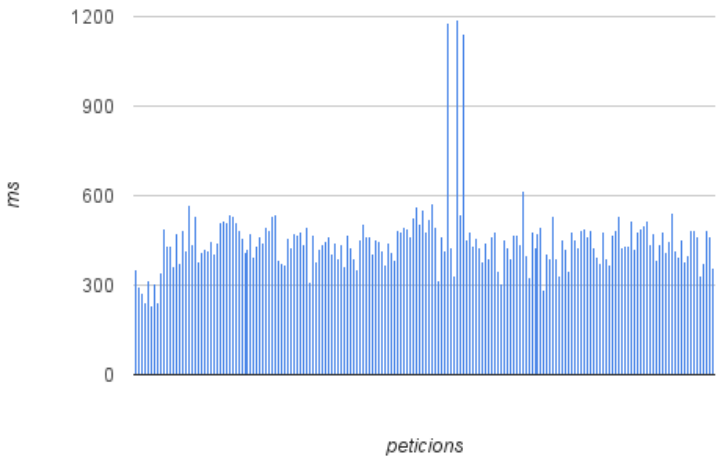


Figura 29 Test amb 10 usuaris.

Test amb 15 usuaris: Temps mitjà = 560ms, peticions per segon: 24,5

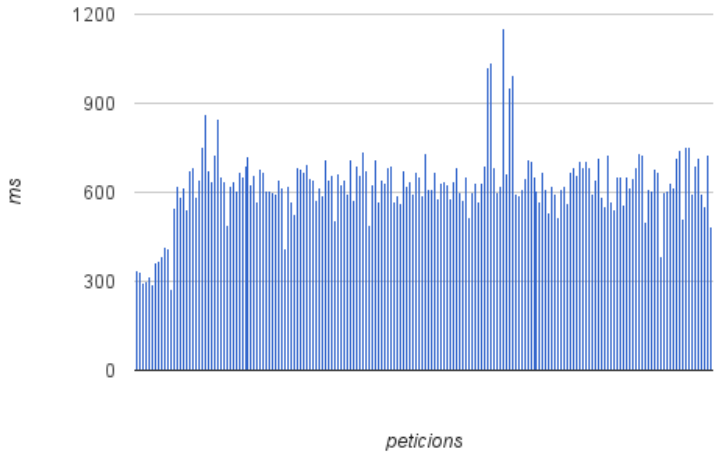


Figura 30 Test amb 15 usuaris.

Test amb 20 usuaris: Temps mitjà = 684ms, peticions per segon: 23,8

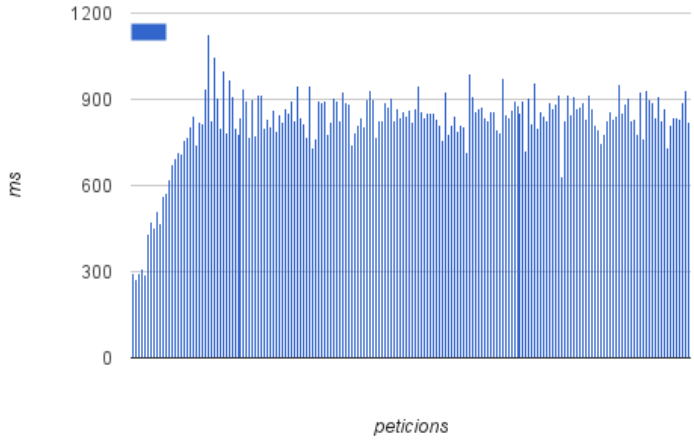


Figura 31 Test amb 20 usuaris.

El primer test, en el que hem simulat 10 usuaris, podem veure que el temps de resposta té una mitja de 437ms, junt amb rendiment de 24 peticions per segon. Veiem doncs que el servidor actual podria atendre 1440 peticions cada minut amb un temps mitjà per resposta acceptable, i al voltant de dos milions al dia, unes xifres que mostren un bon rendiment. Per fer-nos una idea sobre aquests resultats, podem observar que la web LinkedIn[11] té una mitja de 120 cerques per segon. Escalant el nostre sistema amb cinc servidors amb les mateixes característiques que l'actual (servidors molt senzills), més un sisè que realitzes el paper de distribuïdor de la càrrega podríem donar servei a una web de les magnituds de LinkedIn.

Analitzant la resta dels gràfics, que simulen casos amb més usuaris (15, 20), podem observar que el temps mitjà de resposta empitjora a mesura que augmentem el nombre d'usuaris, però en canvi el nombre de peticions per segon es mantenen constants. Això significa que aquest servidor pot respondre un màxim de 10 peticions simultàniament de forma constant sense empitjorar el temps de resposta, i a mesura que anem incrementant el nombre de peticions simultànies el temps de resposta anirà empitjorant.

Cal dir que en cap dels tests el servidor ha deixat de funcionar correctament tot i encara l'estrès provocat pel gran nombre de peticions, mostrant doncs una gran estabilitat. En el cas extrem en el qual hem simulat 100 usuaris realitzant peticions constantment, el servidor ha seguit responent a les peticions realitzant l'expansió de les consultes correctament, això si, amb uns temps mig de resposta al voltant dels 4 segons. Aquests últims resultats, ens mostren que en vers a la possibilitat de rebre un pic de peticions més alt de l'esperat en un moment donat, el sistema respondria a aquestes correctament però amb un temps de resposta alt, fins que el pic de peticions ja hagi sigut atès.

[11] https://www.linkedin.com/

10.4 Integració de Qeast

En aquest apartat veurem els passos que s'han de seguir per a poder integrar Qeast en una web. Des del registre de la web, fins a com fer la petició d'expansió a Qeast.

Com ja hem vist, el primer past consisteix en registrar la web al sistema mitjançant l'API, aquesta seria la crida necessària per registrar una web amb la url "www.test.com" i la contrasenya *testweb:*

- *http://qeast.qeastsearch.com:9000/register?url=*http://test.com/&pass=*testweb*

Al fer el registre de la web, Qeast ens retorna un identificador per a la web (ID = 1585575464). Una vegada tenim l'identificador de la web, li hem d'indicar a Qeast que iniciï el processament de la web, amb la següent crida a l'API:

- *http://qeast.qeastsearch.com:9000/startProcess/ id=1585575464&pass=testweb*

Un cop hem realitzat aquesta crida cal esperar que el sistema finalitzi el procés i estigui llest per a poder servir les peticions d'expansió de consultes. Amb el sistema ja està llest, hem d'integrar Qeast al nostre servidor i així poder realitzar l'expansió de les consultes dels usuaris abans que es realitzin les cerques. Per exemplificar aquesta part hem decidit mostrar un servidor programat en php, ja que és el llenguatge de programació més estès als servidors. Al Snippet 2 podem veure com és l'aspecte que tindria el codi d'un servidor que s'encarrega de realitzar la cerca dins d'una web.

```
1. <?php
2.     $originalQuery = urlencode($_GET['q']);
3.     $results = search($originalQuery);
4.     echo $results;
5. ?>
```

Snippet 2 Exemple de la implementació d'una cerca

A la línia 2 el servidor obté la consulta de l'usuari, que seguidament utilitza a la línia 3 per realitzar la cerca. Finalment a la línia 4 retorna els resultats a l'usuari.

Com ja hem explicat, hem de procurar fer l'expansió de les consultes abans de fer la cerca, en el Snippet 3 mostrem com integrar la crida a Qeast al servidor i utilitzar tots els termes d'expansió que ens retorna per realitzar les cerques.

```
1. <?php
2.     $originalQuery = urlencode($_GET['q']);
3.     $res=file_get_contents("http://qeast.qeastsearch.com:9000
       /queryExpansion/1585575464/ testweb?query=$originalQuery");
4.     $resJson = json_decode($res, true);
5.     $oldEntities = $resJson["old"];
6.     $newEntities = $resJson["new"];
7.     $entities = array_merge($oldEntities, $newEntities);
```

```
8.  $newQuery = implode(" ", $entities);
9.  $results = search($newQuery);
10. echo $results;
11. ?>
```

Snippet 3 Exemple 1 de la implementació d'una cerca utilitzant Qeast

Podem veure com el servidor a la línia 3 i 4 realitza la crida a Qeast per fer l'expansió de la consulta de l'usuari. Seguidament a les línies 5, 6 i 7 junta els termes d'expansió retornats per Qeast amb la consulta de l'usuari a la variable $newQuery. Finalment a la línia 9 realitza la cerca amb $newQuery (conté l'expansió de la consulta) com a consulta.

Capítol 11

Treball futur

A treball futur s'exposaran aquelles funcionalitats o característiques que encara no s'han implementat o dissenyat, i que són importants per aconseguir un producte final complet.

11.1 Sistema de pagament

Com s'ha pogut observar a l'apartat Qeast in action, Enrich Data s'ha convertit en un producte comercial, i per tant aquest tindrà uns preus i uns mètodes de pagament. Actualment, el sistema es troba en una fase beta i és gratuït. Caldrà dissenyar i implementar un sistema de pagament perquè els clients puguin abonar la quantitat fixada.

11.2 Sistema de sincronització automàtica

Quan una web es registra a Qeast, es genera la seva KB segons els continguts que conté la web en el moment del procés, i per tant si s'està donant servei a una plana web que dia a dia va afegint nous continguts, ens trobem amb què els nous continguts no es tindran en compte al fer l'expansió de les, aquest fet suposa un obstacle per fer una expansió de les consultes completa.

Per solucionar-ho s'implementarà un sistema que automàticament es descarregarà els nous continguts que es van afegint a les webs, afegint-los a les KBs pertinents per seguidament refer el procés d'extracció de relacions, ara si amb tots els nous continguts.

El procés d'obtenció dels nous continguts s'implementarà de dues formes diferents, si la web que s'està tractant disposa de canal RSS s'accedirà als nous continguts utilitzant aquesta via, en el cas contrari s'implementarà un sistema automàtic que guardarà les url ja emmagatzemades, i recorrent la web descarregarà el contingut d'aquelles que encara no s'han tractat.

11.3 Contextualització de la consulta

Com s'ha vist a l'apartat del mòdul Query expander, quan el sistema rep una consulta per a realitzar-ne l'expansió, el primer pas consisteix en extreure-li les entitats. Aquest pas

però, pot tindre algunes limitacions.

Quan Dexter està processant un text, tracta d'identificar el seu context per identificar millor les entitats que conté. El context és el que ens ajuda a identificar en casos d'ambigüitat on dues o més entitats s'anomenen igual, a quina està fent referencia el text. Per exemple, el terme "apple" és ambigu, ja que pot fer referencia a la fruita o a la multinacional que dissenya i fabrica productes tecnològics. En un context culinari seleccionaríem la primera opció, en canvi en un context tecnològic seleccionaríem la segona.

Si utilitzar Dexter per identificar les entitats del text "apple", la manca de context fa que sempre identifiqui "apple" com l'entitat que fa referencia a la multinacional. Veient aquest fet, si un usuari web fa la mateixa consulta al blog "www.iamafoodblog.com" (introduït al capítol 10) amb la intenció de buscar receptes amb poma, durant el procés d'expansió aquesta consulta passarà per Dexter, que com hem vist explicat anteriorment, a causa de la falta de context de la consulta sempre ens retornarà l'entitat que fa referencia a la multinacional i no a la fruita. Com a conseqüència, nos es seleccionarien correctament els termes d'expansió i el procés d'expansió serà erroni.

Per solucionar aquest problema s'ha plantejat contextualitzar la cerca afegint uns termes fixos a les consultes que varien segons la web de la petició abans que Dexter les processi. Per exemple, en el cas del blog d'alimentació, podríem afegir els termes "food" i "recipe" per realitzar aquesta contextualització. Una vegada feta la identificació d'entitats, s'haurien d'eliminar aquelles relacionades amb els termes afegits per contextualitzar.

Al utilitzar Dexter sobre la consulta "apple" amb la contextualització proposada anteriorment ("food" i "recipe"), ara sí que Dexter identificat l'entitat "apple" que fa referencia a la fruita.

La via per escollir aquests termes de contextualització encara s'està estudiant. S'hauria d'implementar un procés que per a cada KB analitzés la seva centralitat i s'escollís les entitats més rellevants per utilitzar-les com a context per a les consultes. Un possible algorisme per implementar aquest procés podria ser el PageRank.

11.4 Processat de fitxers

Enrich Data només emmagatzema i processa els continguts dels enllaços de les webs que contenen el text pla en fitxers HTML. S'implementarà una nova funcionalitat que permeti extreure el contingut de fitxers que Enrich Data pot trobar a les webs, com pdfs o arxius .doc, i així abastar un domini de la web més gran i complet.

11.5 Estendre el sistema a altres idiomes

Com es va definir a l'abast del projecte, el producte actual només és funcional per a webs que tenen tot el seu contingut en anglès. En un futur però, Enrich Data podrà processar webs en altres idiomes.

Per implementar aquesta nova funcionalitat cal assolir dues metes, generar un nou graf amb la versió de la Wikipedia de l'idioma que es vol poder processar i adaptar Dexter perquè pugui identificar entitats en texts escrits en aquest idioma.

Capítol 12

Conclusions

Com a objectiu principal del projecte, vam proposar el disseny i desenvolupament d'un sistema que contribuís a millorar les webs, així com l'experiència dels usuaris, mitjançant un sistema d'expansió de consultes que optimitzi els resultats de les cerques. Un cop finalitzat el projecte, cal fer una anàlisi retrospectiva i veure si s'han assolit els objectius que es van fixar.

Actualment Enrich Data és capaç de realitzar tot el procés descrit en aquest document, permetent la realització correcta de les expansions de les consultes a les webs. Els resultats que hem obtingut de les proves d'expansió de consultes són molt bons, mostrant un bon funcionament del sistema desenvolupat. Enrich Data també ha demostrat una gran estabilitat durant els tests d'estrès, i un gran rendiment que junt amb la seva capacitat d'escalar horitzontalment, ens ofereix la possibilitat de donar un servei de qualitat a una gran quantitat de clients.

Podem dir llavors, que hem assolit l'objectiu principal d'aquest projecte, junt amb tots els objectius específics, obtenint com a resultat un producte final més que satisfactori.

També cal assenyalar que, com hem vist al Capítol 11 Treball futur, encara queden funcionalitats per afegir. Aquest és un fet positiu, ja que tot i que Enrich Data ja és un producte funcional, té marge per seguir millorant i creixent com a producte.

Des d'un punt de vista més personal, estic molt satisfet amb el treball realitzat. Considero que la feina s'ha realitzat correctament, i que tots els esforços invertits en aquest projecte han valgut la pena. Durant el transcurs d'aquest projecte, he crescut com a professional, aprenent des de noves nocions tècniques fins a habilitats comunicatives i de treball en equip.

D'altra banda, aquest projecte ha assentat tots els coneixements que he anat aprenent durant el grau d'Enginyeria Informàtica, i m'ha permès conèixer de prop el món de la recerca. Això, de fet, m'ha motivat per continuar els meus estudis amb la voluntat de començar un Màster.

En conclusió, tant el camí recorregut com el resultat final del projecte, han estat altament satisfactoris per l'experiència viscuda i pel producte final aconseguit.

Bibliografia

1. **Norbert Martínez-Bazan.** *Efficient graph management based on bitmap indices.* s.l. : IDEAS 2012: 110-119, 2012.

2. **Varis autors.** MongoDB. 2009. https://www.mongodb.com/.

3. **Robert Griesemer et al.** Go. https://golang.org/.

4. **Jason van Zyl.** Maven. 2002. https://maven.apache.org/.

5. **Varis autors.** Revel Framework. https://revel.github.io/.

6. **Matt Mullenweg et al.** Wordpress. 27 / 5 / 2003. https://es.wordpress.com/.

7. **Joan Guisado-Gámez et al.** *ENRICH: A Query Rewriting Service Powered by Wikipedia Graph Structure.* s.l. : Tenth International AAAI Conference on Web and Social Media, 2016.

8. **DAMA-UPC.** Wikiparser. 2016. https://github.com/DAMA-UPC/WikiParser/.

9. **Joan Guisado-Gámez et al.** *Understanding Graph Structure of Wikipedia for Query Expansion.* Proceedings of the GRADES'15, 2015.

10. **Joan Guisado-Gámez et al.** *Query Expansion via structural motifs in Wikipedia Graph.* Barcelona 12 / 05 / 2016. arXiv preprint arXiv:1602.07217 .

11. **D. Ceccarelli.** *Dexter: an Open Source Framework for Entity Linking.* Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR), 2013 .

Apèndix

Article presentat al WikiWorkshop

En el marc d'aquest projecte hem publicat un article en un *peer reviewed workshop* anomenat WikiWorkshop. Emmarcat en les conferències "World Wide Web" i "International conference on Web and Social Media", WikiWorkshop s'especialitza en l'ús de la Wikipedia per al desenvolupament de projectes tecnològics. A continuació adjuntem l'article que vam presentar i que va ser acceptat i presentat i que resumeix els objectius d'Enrich Data.

The Workshops of the Tenth International AAAI Conference on Web and Social Media
Wiki: Technical Report WS-16-17

ENRICH: A Query Rewriting Service Powered by Wikipedia Graph Structure — Extended Abstract

Joan Guisado-Gámez
DAMA-UPC
Universitat Politècnica de Catalunya
joan@ac.upc.edu

David Tamayo-Domènech
DAMA-UPC
Universitat Politècnica de Catalunya
tamayo@ac.upc.edu

Jordi Urmeneta
Sparsity-Technologies
Barcelona
urmeneta@sparsity-technologies.com

Josep-Lluís Larriba-Pey
DAMA-UPC
Universitat Politècnica de Catalunya
larri@ac.upc.edu

Abstract

The search for relevant information in websites can be very frustrating for users who, unintentionally, use too general or inappropriate keywords to express their requests. To overcome this situation, query rewriting techniques aim at transforming the users requests to better describe the real intent of the users. However, to the best of our knowledge, current search tools either are too generic or require resources not available for everyone such as query log processors, natural language engines, etc. To supply this need, we present ENRICH, which is a query rewriting cloud service that is automatically tailored to each website and it is powered by an available, accessible and open resource: Wikipedia.

1 Introduction

Nowadays, all the institutions, most of large and small business and many people have their own websites, as it has become one of the most common ways to disseminate information. However, the process of searching for information in each of those sites can be a tedious task for users who often obtain a "No results found" message. It may happen that, despite the message, the site has information about the topic the user is looking for, but the vocabulary used in the website is different from the user's. This phenomenon is called **vocabulary mismatch** and it is common in the usage of natural language processes. Also, the **topic inexperience** of the users, which is caused by the lack of familiarity with the vocabulary, entails that not all the interesting documents of the site are retrieved.

Query rewriting techniques aim at improving the results achieved by the user search by means of introducing new terms, commonly called **expansion features** and/or removing terms from the original query. Thus, the challenge is to select those expansion features that are capable of improving the results the most. However, it is difficult for institutions, small business or people to have the technology to implement such techniques. As a response to this need, specialized companies in information retrieval have become third parties that offer search solutions. For example, Google Search Appliance (GSA) (Google^{TM} 2016) is an integrated, all-in-one hardware and software, that provides Google search technology for organizations. However, this technology is thought and designed for large organizations that can afford it. Moreover, for GSA to exploit its full potential, and to retrieve qualitative results, it is suggested to **manually** create files of customized expansion terms for the specific vocabulary of the site[1].

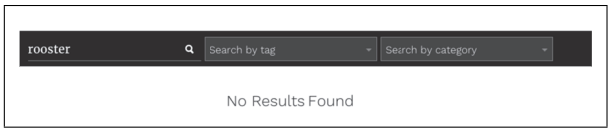
Previous research (Guisado-Gámez, Dominguez-Sal, and Larriba-Pey 2014; Guisado-Gámez and Prat-Pérez 2015; Guisado-Gámez, Prat-Pérez, and Larriba-Pey 2016) has shown that the graph structure of Wikipedia, which consists of articles and categories related to each other, encodes relevant information, which allows extracting reliable expansion features. Thanks to the support of the EU-Tetracom initiative, in this paper we present ENRICH, which is a collaborative task between academia and industry to take advantage of previous research findings. ENRICH is a query rewriting system service that specializes its expansions for each particular website. It uses Wikipedia as a generic knowledge base (KB) out of which it derives a website-specific knowledge base (WS-KB), the structure of which is exploited to identify strongly related concepts that are good candidates to be used as expansion features.

The rest of the paper is organized as follows. In Section 2 we give an overview of ENRICH. In Section 3 we provide details about the architecture behind ENRICH. Finally, in Section 4 we conclude and outline our future work.

2 ENRICH Overview

The main goal of ENRICH is to improve the search experience of users, offering a query rewriting service in the cloud that is based on Wikipedia and really easy for webmasters to integrate it in their sites.

[1] https://goo.gl/oQ2YoR.



(a) User query: `rooster`.

(b) ENRICH query: `chicken`.

Figure 1: ENRICH in http://iamafoodblog.com.

ENRICH specializes its expansions for each site as opposed to general query rewriting techniques, which offer generic solutions independently of the website topic and vocabulary. For that purpose, ENRICH analyzes each website and uses Wikipedia to identify its entities, which are defined as the real world concepts. It also identifies the way they are referred in the website. Notice that the same entity can have several names, for example, *car*, *auto*, *automobile* are alternative names for the same entity. We call the set of entities that appear in the website, **website entities**, and their names (those that are used in the website), **appearance names**.

Notice that search engines only will retrieve documents if the user's query matches any of the appearance names. To increase the hit rate of the search engine, ENRICH **automatically** builds a website-customized rewriting file that allows translating the user queries into a set of appearance names. In order to do that, ENRICH follows two strategies: First, for each website entity, it finds the rest of its names besides its appearance names. Second, for each website entity it finds a set of strongly related entities, so that, their names can be translated into the appearance names of the website entity. To illustrate this second strategy, imagine the scenario in which *car* is a website entity, but *vehicle* is not (i.e. there is no website page in which it appears). Since *car* and *vehicle* represent two strongly related entities, ENRICH would translate the latter into the former in a way that the search engine could retrieve car-about pages. In order to follow this strategy, ENRICH uses Wikipedia to build, for each website, a specific knowledge base (WS-KB). Then, the structure of the WS-KB is analyzed to identify strongly related entities.

As an example of ENRICH capabilities, we have applied it to http://iamafoodblog.com. We show two examples of ENRICH rewriting a query for this site:

- **Query 1 (Q1)**: `Rooster`
  ENRICH query: `Chicken`.
- **Query 2 (Q2)**: `Sausage`
  ENRICH query: `Sausage`, `hot dog`, `chorizo`, `chinese sausage`, `sausage roll`, `merguez`,...

In Q1 the user is looking for posts that talk about `roosters`. However, the website does not contain any post that uses that particular term, therefore, the search engine returns a "No results found" message as depicted in Figure 1a. Thanks to ENRICH, the query is rewritten as `chicken`, which allows the search engine to return 130 posts as shown in Figure 1b. This example shows that ENRICH is capable of overcoming the vocabulary mismatch problem. In Q2, the user is looking for posts talking about `sausages`. Although there are up to 31 posts that talk about sausages, the results can be improved if they are combined with those obtained by more specific queries, such as `hot dog`, `chorizo`, which is a Spanish sausage, and `merguez`, which is a typical sausage from Maghreb, etc. This situation shows a scenario of topic inexperience that ENRICH is capable of overcoming by adding strongly related website entities' names.

Notice that the use of ENRICH is completely transparent for website users, who are not conscious, in any case, of the system working for the particular website they are querying. A user would simply introduce the query in a typical search box, as the ones depicted in Figure 1, and the website would return the results. Nonetheless, to make ENRICH work properly, the webmaster has to modify the website code of its site to integrate it. The modifications are minor and consist in capturing the user's query and sending it to ENRICH via a REST API. Once ENRICH receives a request, it identifies the entities within the user's query, accesses the web-customized rewriting file, and returns the corresponding appearance names. The result is in the form of a JSON text that contains 2 fields, the appearance names that are explicitly in the user's query, and the set of appearance names that are introduced due to the analysis of its WS-KB. It is the responsibility of the webmaster to use the names in the returned JSON to send the rewritten query to the search engine.

In Snippet 1 we show a piece of the code that webmasters could use to capture the user's query and to send it to ENRICH. The user's query is the `input` of the function. In line 3, ENRICH is called by specifying its URL (`enrichserver`), the website id (`11346`) and its password (`pwd`). Once the function returns the rewritten query, line 7, the expansion features (`finalQuery.expFeat`) are sent to the search engine, in line 7.

```
1 $scope.queryExpansion = function(input){
2   $http({ method:'GET',
3         url: 'https://enrichserver/queryExpansion/11346/pwd',
4         params: {query:input}}).then(
5         function successCallback(response){
6           $scope.finalQuery = response;
7           $scope.search($scope.$finalQuery.expFeat); });
8   }
```

Snippet 1: JavaScript function that calls ENRICH.

3 ENRICH Architecture

In Figure 2 we schematically show the architecture behind ENRICH. We distinguish three main blocks, which consist in i) loading the Wikipedia graph, ii) building the WS-KB and iii) analyzing it. In the rest of this section we explain in detail each of these blocks.

3.1 Wikipedia Graph Load

The goal of this block is to load Wikipedia into a Graph Database Management System (GDBMS) to easily exploit

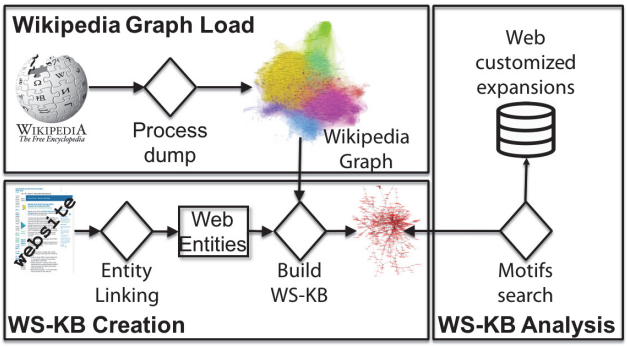


Figure 2: ENRICH architecture.

its structural properties. For that purpose, we need to parse the Wikipedia dump to obtain i) article ids and $titles$, ii) category ids and $names$, iii) article redirections, iv) links among articles, v) links among categories and vi) links among articles and categories. For that purpose, we have developed WikiParser [2], which is a tool that parses the English Wikipedia to CSV. It requires i) pages-articles.xml, ii) page.sql and iii) categorylinks.sql Wikipedia's dump files to create 6 CSV files, each of which contains the information previously described. Notice that since ENRICH is based on Wikipedia's structural properties, the body of the articles is not required.

Then, we use the files to load Wikipedia into Sparksee (Martínez-Bazan and et al. 2012), which is a GDBMS that allows performing complex operations efficiently. To load the data, we discard all the relations with the hidden categories, which are a special kind of categories that are concerned with the maintenance of Wikipedia, rather than being part of the content of the encyclopedia. In our experience, the English Wikipedia, without the body of the articles, loaded in Sparksee requires 11Gb of disk.

This process is carried on whenever it is needed it, depending on the updates of Wikipedia that affect its overall structure. However, despite of the high frequency of updates in Wikipedia articles, and although, a more exhaustive analysis is required, we believe that the main structure of Wikipedia does not change dramatically in each dump.

3.2 WS-KB Creation

This block consists in building the specific knowledge base for each site and identifying the strongly related articles. Notice that although Wikipedia acts as a generic KB in the form of a graph, each WS-KB is a subgraph of Wikipedia that includes the web's topics.

In order to create the WS-KB, first, we need to identify the website entities, and match them with the corresponding articles of the Wikipedia graph. Note that an entity is a concept, and its materialization in the Wikipedia graph is an article. We call the materialization of the website entities in the graph, **website articles**. For that purpose, we use Dexter (Ceccarelli et al. 2013) which is an open source project that actually recognizes entities in a given text and matches them with Wikipedia articles, providing its corresponding id, for each website entity. According to our experiments, which consisted in identifying and linking the entities in the queries of 3 datasets (ImageCLEF, CHiC 2012 and CHiC 2013), Dexter successes in 96% of the occasions. In this process, we also annotate its appearance name.

Second, although, the website articles constitute the core of the WS-KB, it is required that it contains more articles and categories, otherwise, we could not relate the appearance names of the website entities to other entities that do not appear in the website and their names. This would prevent ENRICH from overcoming the vocabulary mismatch problem. Our current proposal consists in building the WS-KB with the website articles, their redirects [3], their linked articles, their categories and the articles that belong to those categories. Note that we use the Wikipedia graph to identify the edges among the nodes and add them into the WS-KB.

The process of building the WS-KB is done the first time a webmaster installs ENRICH and each time that he/she considers that it is required to modify the web-customized rewriting file.

3.3 WS-KB Analysis

To identify the tightly linked articles in Wikipedia, we base our proposal on (Guisado-Gámez and Prat-Pérez 2015), where we analyzed relevant structures in Wikipedia that allow relating those articles that are close semantically with no need of any linguistic analysis. The analysis revealed that cycles (defined as a closed sequence of nodes, either articles or categories, with at least one edge among each pair of consecutive nodes) were important and relevant to relate them. Summarizing the characteristics that let us differentiate good from bad cycles, we have that:

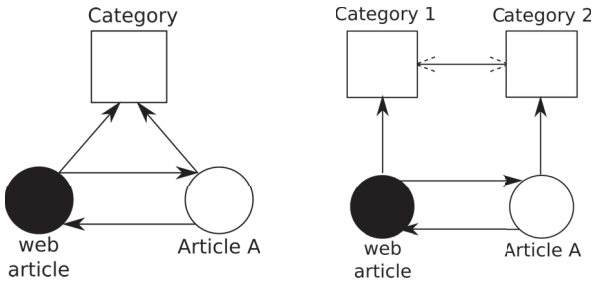
- Cycles of length 2 are not reliable.
- Cycles of length 3, 4 and 5 are to be trusted to reach articles that are strongly related with the website articles.
- Around a third of the nodes of cycles have to be categories. This ratio is expected to increase beyond the cycles of length 5.
- The expansion features obtained through the articles of dense cycles are capable of leading to better results.

Based on these characteristics we propose the motifs depicted in Figures 3a and 3b, which are based on cycles of length 3 and 4 respectively. The motif depicted in Figure 3a is called, from now on, **triangular motif**, while the one depicted in Figure 3b is called **square motif**. In the figures, the square nodes are categories, while round nodes are articles. The black round node is a website article, while the white round node is an article A, a new article selected as it forms a motif with the website article.

In the triangular motif we force the website article to be doubly linked with article A. That means that the website

[2] https://github.com/DAMA-UPC/WikiParser

[3] If the website article is a main article, we add all the redirects of this article, if it is a redirect articles, we add the corresponding main article, and also all its redirects.



(a) Triangular motif. (b) Square motif.

Figure 3: Expansion motifs.

article actually links, in Wikipedia, to A, and A links, reciprocally, to the website article. Moreover, article A must belong to, at least, one of the categories of the website article. This structure guarantees a strong relation between the website article and article A. In the square motif of Figure 3b, the website article and article A must be also doubly linked. However, compared to the triangular motif, it is just required that at least one of the categories of the website article is linked with one of the categories of A, or *vice versa* (depicted with a dashed arrow in Figure 3b). This pattern still guarantees a strong relation between the website article and article A, but it is not as restrictive as the one represented in Figure 3a. Both patterns are chosen because they make sense from an intuitive point of view. It is expected that doubly linked articles that are also connected through categories, are also close semantically (although the system has not done any kind of semantic analysis), and the title of one can serve as an expansion feature of the other. Also because these cycles fulfill the edge density and ratio of categories requirements. To decide the length of the cycles that we base the motifs on, we ignore those of length 2, as they resulted not to be trustful to identify strongly related articles. Larger cycles, as those with a length larger or equal to 5, have been also avoided for performance reasons. The traversal of larger cycles expands too much the search space in the WS-KB, and would make it difficult to identify them in a reasonable time for query rewriting processes. Moreover, our previous results in (Guisado-Gámez, Prat-Pérez, and Larriba-Pey 2016) show that the motifs depicted in Figure 3 allow up to 150% improvement.

Given a website article, ENRICH identifies all its strongly related articles as those other articles that share, at least, one motif. This allows relating its appearance names (which we annotated at the beginning of the process) represented by a website article, with a set of articles, each of which have a title, and that may have several redirect articles. The article and redirects titles are used as the set of names recognized by ENRICH and that are translated into the appearance names. This constitutes the web-customized rewriting file.

Notice that in order to fulfill the performance requirements of a system like ENRICH, the access to the rewriting file must be done as fast as possible. For that purpose, we load this file into an indexed and in-memory structure that relates each recognized name (all the names of all the entities represented in the WS-KB) with the corresponding appearance name. Notice that the structure required only to represent the entities in Q1 and Q2 would consist of 10 entities, 138 recognized names (all the names of these entities) and 7 appearance names.

The process of analyzing the WS-KB is done always after the WS-KB is created, under webamaster's demand.

4 Conclusion & Future Work

In this paper we have presented the main ideas behind ENRICH, a query rewriting system. ENRICH differs from other current software in two main aspects: first, it specializes its query rewrites for each website, second, it uses the Wikipedia structure to identify the expansion features taking into account the website vocabulary. Most of the research regarding to Wikipedia is based on improving the methods for extracting and using its content. However, in previous research, we showed that exploiting exclusively the structure of Wikipedia, without using any kind of language analysis, allows achieving remarkable results.

ENRICH is still in its development phase, thus, we cannot provide any experimental result in this paper. Nonetheless, in our previous results (Guisado-Gámez, Prat-Pérez, and Larriba-Pey 2016), which did not include the rewriting index, we have shown that queries are rewritten in less than 0.2s, and achieved up to 150% improvements. We expect to drastically reduce the rewriting time due to the index. We also expect to improve the results due to the WS-KB tailored to each particular website.

Acknowledgments

The members of DAMA-UPC thank the Ministry of Science and Innovation of Spain and Generalitat de Catalunya, for grant numbers TIN2009-14560-C03-03 and SGR2014-890 respectively. The members of DAMA-UPC and Sparsity-Technologies also thank the EU-Tetracom for grant agreement number 609491.

References

Ceccarelli, D.; Lucchese, C.; Orlando, S.; Perego, R.; and Trani, S. 2013. Dexter: an open source framework for entity linking. In *ESAIR*, 17–20.

Google^{TM}. 2016. *Google Search Appliance, http://google.com/enterprise/gsa*.

Guisado-Gámez, J., and Prat-Pérez, A. 2015. Understanding graph structure of wikipedia for query expansion. In *GRADES*, 6:1–6:6.

Guisado-Gámez, J.; Dominguez-Sal, D.; and Larriba-Pey, J. 2014. Massive query expansion by exploiting graph knowledge bases for image retrieval. In *ICMR*, 33.

Guisado-Gámez, J.; Prat-Pérez, A.; and Larriba-Pey, J. 2016. Query expansion via structural motifs in wikipedia graph. *CoRR* abs/1602.07217.

Martínez-Bazan, N., and et al. 2012. Efficient graph management based on bitmap indices. In *IDEAS*, 110–119.