

# UNIVERSIDAD DE CUENCA



## FACULTAD DE INGENIERÍA

### ESCUELA DE INGENIERÍA DE SISTEMAS

“Plataforma para la Anotación Semántica Automática de Servicios Web RESTful sobre un Bus de Servicios”

Tesis previa a la obtención del  
Título de Ingeniero de Sistemas

Autores:

José Enrique Ortiz Vivar.  
José Luis Segarra Flores.

Director:

Ing. Víctor Hugo Saquicela Galarza, PhD.

Cuenca - Ecuador  
2015

---

## Resumen

---

Hoy en día, los Servicios Web son ampliamente usados en un gran número de aplicaciones en la Web. La mayoría de dichas aplicaciones son servicios de información sobre: clima, deportes, noticias entre otras, que se basan en el uso de la arquitectura REST. Debido a la masificación que los Servicios Web han alcanzado, es evidente la necesidad de mecanismos más sofisticados para su gestión y explotación, por lo tanto, actividades como descubrimiento, búsqueda y composición de Servicios Web han llegado a ser tareas cotidianas entre los desarrolladores. Sin embargo, la implementación de estas tareas trae algunos problemas causados por la falta de automatización de sus procesos, los requieren de una alta intervención humana.

Actualmente, el principal enfoque para solucionar los problemas causados por la falta de automatización en las tareas relacionadas con la gestión de Servicios Web es la incorporación de anotaciones semánticas. Este enfoque tiene por objetivo facilitar a los computadores a interpretar y entender la información de los Servicios Web, para que así las tareas antes mencionadas (descubrimiento, búsqueda y composición) puedan ser automatizadas. No obstante, la mayoría de los procesos de anotación propuestos hoy aún requieren de procedimientos manuales. En este contexto, la presente tesis describe el desarrollo de una plataforma para anotación semántica automática de Servicios Web, plataforma que permite la generación de anotaciones semánticas con una reducida intervención humana, a través del desarrollo de un componente permite la selección automática de ontología de acuerdo al dominio de los Servicios Web.

**Palabras clave:** SPARQL, RESTful, Servicios Web, Web Semántica, RDF, Anotación Semántica, Linked Data, Bus de Servicios, Patrones de integración, Plataforma de anotación, Selección de ontologías, Automatización.

José Enrique Ortiz Vivar  
José Luis Segarra Flores

---

## Abstract

---

Nowadays, Web Services are widely used so that there is an immense collection of applications on the Web. Most of such applications, like information services about: weather, sports, news, among and others, rely on the use the REST architecture. Due to the massification that Web Services have reached, it is evident that more sophisticated mechanisms for managing and exploitation them are needed therefore, activities of web services such as discovery, searching and composition have become daily tasks among developers. However, some tasks implementation brings some obstacles caused by the lack of automatization within its processes, which requires high human intervention.

Currently, the main approach in solving the problem proposed by the lack of automation tasks related to the management of Web Services corresponds to the incorporation of semantic annotations. This approach aims to help computers to interpret and understand the information of Web Services, so that the aforementioned tasks (discovery, searching and composition) can be automated. Nonetheless, most of semantic annotations approaches require manual procedures up to date. Within this context, this thesis work describes the development of a platform for Web Services automatic semantic annotation, a platform that allows generating semantic annotations with reduced human intervention through the development of a component that allows the automatic selection of ontologies according to the Web Services domain.

**Keywords:** SPARQL, RESTful, Web Services, Semantic Web, RDF, Semantic Annotation, Enterprise Service Bus, Enterprise Integration Patterns, Annotation Platform, Ontologies Selection, Automatization.

José Enrique Ortiz Viver  
José Luis Segarra Flores

---

# Índice General

---

Resumen . . . . .	2
Abstract . . . . .	3
Índice General . . . . .	4
Índice de figuras. . . . .	8
Índice de tablas. . . . .	10
<b>1. Introducción</b>	<b>19</b>
1.1. Antecedentes . . . . .	19
1.2. Planteamiento del problema . . . . .	20
1.3. Justificación . . . . .	20
1.4. Objetivos . . . . .	22
1.4.1. Objetivo general . . . . .	22
1.4.2. Objetivos específicos . . . . .	22
1.5. Alcance . . . . .	22
1.6. Estructura del documento . . . . .	23
<b>2. Marco teórico</b>	<b>24</b>
2.1. Servicios Web . . . . .	24
2.1.1. Servicios SOAP . . . . .	26
2.1.1.1. Mensajes SOAP . . . . .	27
2.1.1.2. WSDL . . . . .	27
2.1.1.3. UDDI . . . . .	28
2.1.1.4. Ejemplo mensaje SOAP . . . . .	28
2.1.2. Servicios REST . . . . .	29
2.1.2.1. Descripción del servicio REST . . . . .	30
2.1.2.2. Ejemplo de llamada a un servicio REST . . . . .	30
2.1.3. Servicios Web REST o Servicios Web SOAP . . . . .	31
2.1.4. Composición de Servicios . . . . .	32
2.2. Web Semántica . . . . .	33
2.2.1. Antecedentes . . . . .	33
2.2.2. Definición de Web Semántica . . . . .	34
2.2.3. Ontologías . . . . .	35
2.2.4. Lenguajes ontológicos de la Web Semántica . . . . .	36
2.2.4.1. RDF . . . . .	37
2.2.4.2. RDF Schema . . . . .	39

2.2.4.3. OWL . . . . .	41
2.3. SPARQL . . . . .	41
2.3.1. SPARQL Endpoint . . . . .	42
2.4. Estado del arte de la descripción semántica de Servicios Web . . .	42
2.4.1. Lenguajes de anotación de Servicios Web . . . . .	43
2.4.1.1. Lenguajes Sintácticos . . . . .	44
2.4.1.2. Lenguajes Semánticos . . . . .	45
2.4.2. Anotación Semántica de Servicios Web . . . . .	47
2.4.2.1. Anotación Manual . . . . .	47
2.4.2.2. Anotación Asistida . . . . .	48
2.4.2.3. Anotación Automática . . . . .	48
2.5. Tecnologías para la Búsqueda y Selección de Ontologías . . . . .	50
2.5.1. Motores de Búsqueda Semánticos de Ontologías . . . . .	51
2.5.2. Técnicas de búsqueda de ontologías basadas en análisis de texto . . . . .	52
2.5.2.1. Emparejamiento entre textos . . . . .	53
2.5.2.2. <i>Clustering</i> de texto . . . . .	54
2.5.2.2.1. Preprocesamiento de Texto . . . . .	54
2.5.2.2.2. Métricas de los clústeres . . . . .	55
2.5.2.2.3. Técnicas de clústeres . . . . .	55
2.5.3. MySQL FullText . . . . .	57
2.5.3.1. Arquitectura general . . . . .	58
2.5.3.2. Ranking en MySQL Fulltext . . . . .	58
2.5.3.3. Búsquedas con MySQL Fulltext . . . . .	60
2.5.4. Apache Solr . . . . .	60
2.5.4.1. Arquitectura General Solr . . . . .	61
2.5.4.2. Indexado y representación de los datos . . . . .	63
2.5.4.3. Ranking en Solr . . . . .	63
2.5.4.4. Búsqueda en Solr . . . . .	64
2.6. Herramientas de integración . . . . .	64
2.6.1. Integración de Software en Ambientes Empresariales . . .	65
2.6.1.1. Arquitectura Orientada a Servicios . . . . .	66
2.6.1.2. Integración de Aplicaciones Empresariales (EIA) .	67
2.6.1.3. Bus de Servicios Empresarial . . . . .	67
2.6.2. Apache ServiceMix como Framework de Desarrollo . . . .	69
2.6.2.1. Apache ServiceMix . . . . .	69
2.6.2.2. Arquitectura de Apache ServiceMix . . . . .	70
2.7. Patrones de integración para el diseño de la solución . . . . .	71
2.7.1. Antecedentes . . . . .	71
2.7.2. Patrones de integración . . . . .	72
2.7.3. Características de los Patrones de Integración . . . . .	73

<b>3. Proceso de Anotación Semántica</b>	<b>76</b>
3.1. Modelamiento del proceso de anotación de Servicios Web . . . . .	77
3.1.1. Adopción del proceso de Anotación Semántica . . . . .	77
3.1.2. Definición y adaptación de las etapas del proceso de anotación	79
3.2. Implementación del proceso de anotación sobre un Bus de Servicios	83
3.2.1. Análisis y selección de las herramientas . . . . .	83
3.2.1.1. Bus de Servicios . . . . .	84
3.2.1.2. Base de datos . . . . .	84
3.2.1.3. Servidor de Aplicaciones Web . . . . .	85
3.2.2. Descripción de componentes . . . . .	86
3.2.2.1. Componente de invocación . . . . .	86
3.2.2.2. Componente para la descripción sintáctica . . . . .	87
3.2.2.3. Componente de enriquecimiento de términos (Sugerencias, Sinónimos) . . . . .	89
3.2.2.4. Componente de selección de ontologías . . . . .	90
3.2.2.5. Componente de <i>Matching</i> o emparejamiento . . . . .	91
3.2.2.6. Componente de validación . . . . .	94
3.2.2.6.1. Validación de entradas . . . . .	94
3.2.2.6.2. Validación de salidas . . . . .	97
3.2.2.7. Componente de persistencia . . . . .	99
3.2.3. Desarrollo de la interfaz gráfica para la plataforma de anotación . . . . .	100
3.2.3.1. Análisis de funcionalidades . . . . .	100
3.2.3.2. Consideraciones de implementación . . . . .	105
<b>4. Selección automática de ontologías</b>	<b>106</b>
4.1. Planteamiento de métodos de búsqueda de ontologías . . . . .	107
4.2. Búsqueda de ontologías en repositorios externos . . . . .	107
4.2.1. Elementos para la organización de datos en el repositorio .	108
4.2.2. Búsqueda directa utilizando Datahub . . . . .	109
4.2.3. Búsquedas por Datahub mediante la adaptación de entradas	112
4.2.3.1. Búsqueda por dominio . . . . .	113
4.2.3.2. Búsqueda por palabras clave . . . . .	115
4.2.3.3. Búsqueda por conceptos . . . . .	116
4.2.3.4. Búsqueda por extracción de entidades . . . . .	118
4.2.4. Búsqueda por dominio, usando medidas de relación semántica	121
4.3. Buscadores semánticos . . . . .	125
4.3.1. WATSON . . . . .	126
4.3.2. Falcons . . . . .	130
4.4. Búsqueda dentro de un repositorio local . . . . .	134
4.4.1. Creación de un repositorio local . . . . .	135
4.4.1.1. Fuente de las ontologías a utilizar . . . . .	135
4.4.1.2. Diseño del repositorio . . . . .	136
4.4.1.3. Extracción del esquema (TBOX) de las ontologías	138
4.4.1.4. Almacenamiento de las ontologías en el repositorio	140



4.4.1.5. Proceso de Carga y actualización del repositorio . . . . .	142
4.4.1.6. Implementación . . . . .	143
4.4.2. Implementación de los métodos de búsquedas sobre el repositorio . . . . .	144
4.4.2.1. Selección de ontologías a través del DBMS . . . . .	145
4.4.2.2. Selección de ontologías mediante un motor de búsqueda de textos . . . . .	147
4.4.2.3. Selección de ontologías a través de técnicas de <i>matching</i> personalizadas . . . . .	149
4.4.3. Servicio Web para el acceso a los métodos de búsqueda . . . . .	156
4.4.3.1. Parámetros del Servicio Web . . . . .	157
4.4.3.2. Diseño interno . . . . .	159
4.4.3.3. Tabla para almacenamiento temporal de resultados (Caché) . . . . .	159
4.5. Resumen de los métodos tratados para la búsqueda y selección de ontologías . . . . .	161
<b>5. Evaluación y Resultados</b>	<b>163</b>
5.1. Métricas de evaluación . . . . .	163
5.1.1. Matriz de confusión multiclasificación . . . . .	165
5.2. Evaluación . . . . .	168
5.2.1. Proceso de evaluación . . . . .	168
5.2.2. Ejecución de la evaluación . . . . .	169
5.2.2.1. Ontologías exclusivamente de dominio específico .	170
5.2.2.2. Ontologías de dominio específico y multidominio	171
5.2.3. Interpretación de resultados . . . . .	171
5.2.3.1. Métodos de búsqueda basados en Solr . . . . .	172
5.2.3.2. Métodos de búsqueda basados en MySQL FullText	172
5.2.3.3. Métodos de búsqueda basados en emparejamiento simple y con medidas de similitud . . . . .	173
5.2.3.4. Métodos de búsqueda <i>clustering</i> . . . . .	173
5.2.4. Resultados del proceso de anotación semántica de Servicios Web . . . . .	173
<b>6. Conclusiones</b>	<b>177</b>
<b>A. Anexos</b>	<b>180</b>
<b>B. Archivo de esquema para Solr.</b>	<b>185</b>
<b>Acrónimos</b>	<b>188</b>
<b>Bibliografía.</b>	<b>196</b>

---

## Índice de figuras.

---

2.1. Elementos de un WS SOAP [23] . . . . .	26
2.2. Nivel de interés de búsquedas de servicios en el tiempo . . . . .	32
2.3. Porcentaje de Servicios Web contenidos en programmableWeb . .	33
2.4. Ejemplo de representación por grafos de una ontología . . . . .	37
2.5. Representación de una tripleta [80] . . . . .	38
2.6. Ejemplificación de un grafo RDF . . . . .	39
2.7. Representación con grafos de un ejemplo en RDFS . . . . .	40
2.8. Interfaz SWEET [45] . . . . .	49
2.9. Etapas del sistema de anotación automática [21] . . . . .	49
2.10. Representación de la clusterización por jerarquía [74] . . . . .	56
2.11. Representación de la clusterización por agrupación [8] . . . . .	57
2.12. Arquitectura general de MySQL Fulltext [59] . . . . .	59
2.13. Arquitectura general de Solr [28] . . . . .	62
2.14. Arquitectura ServiceMix . . . . .	71
2.15. Vista de los componentes en Apache ServiceMix y su funcionalidad [66] . . . . .	72
2.16. Varios patrones de integración [36] . . . . .	75
3.1. Modelo de objetos para la descripción sintáctica de un Servicio Web	78
3.2. Modelo de objetos para descripción de anotaciones . . . . .	78
3.3. Arquitectura general de la plataforma planteada . . . . .	79
3.6. Modelo de objetos propuesto . . . . .	80
3.4. (1/2)Proceso de anotación expresado con patrones de integración.	81
3.5. (2/2)Proceso de anotación expresado con patrones de integración.	82
3.7. Ejemplo de ejecución del componente de invocación . . . . .	87
3.8. Ejemplo de extracción de parámetros . . . . .	88
3.9. Enriquecimiento por sinónimos y sugerencias [67] . . . . .	89
3.10. Ejemplo de ejecución del componente de selección de ontologías .	91
3.11. Ejemplo de ejecución del componente de emparejamiento . . . .	93
3.12. Ejemplo de ejecución del componente de validación de entradas .	97
3.13. Ejemplo de ejecución del componente de validación de salidas .	98
3.14. Captura de la funcionalidad para iniciar el proceso de anotación .	101
3.15. Captura de la funcionalidad de monitoreo del proceso de anotación	103
3.16. Captura de la funcionalidad de visualización de Servicios Anotados	104



3.17. Captura de la funcionalidad de búsqueda de ontologías . . . . .	105
4.1. Resultados sobre GATE Developer . . . . .	120
4.2. Resultado obtenido con Falcons . . . . .	132
4.3. Modelo relacional: Esquema simplificado . . . . .	137
4.4. Modelo relacional: Texto Simple . . . . .	137
4.5. Ejemplo de instanciación de clases con RDF . . . . .	138
4.6. Ejemplo de representación: Concatenación de conceptos . . . . .	140
4.7. Ejemplo de representación: División de términos . . . . .	141
4.8. Ejemplo de representación: Palabras únicas . . . . .	141
4.9. Esquema del repositorio . . . . .	143
4.10. Métodos de búsqueda con sus fuentes de información . . . . .	145
4.11. Consultas sobre MySQL . . . . .	146
4.12. Esquema del proceso de clusterización utilizando la librería de Weka	153
4.13. Pasos de la variación del Bisecting k-means . . . . .	156
4.14. Esquema de búsqueda basada en textos . . . . .	156
4.15. Diseño interno del Servicio Web . . . . .	159
4.16. Tabla de resultados (Caché) . . . . .	160
5.1. Ontologías de dominio específico: Representación en barras de los resultados . . . . .	170
5.2. Ontologías de dominio específico y multidominio: Representación en barras de los resultados . . . . .	171
5.3. Captura de los parámetros ingresados en la plataforma . . . . .	174
5.4. Gráfica de barras de los resultados del proceso de anotación . . . . .	175

---

# Índice de tablas.

---

2.1. Comparativa REST vs SOAP . . . . .	31
2.2. Métricas de similitud [9] . . . . .	53
2.3. Métricas de similitud y distancia comunes en el ámbito de clusterización . . . . .	55
2.4. Ejemplo de caracteres especiales usados en las búsquedas de Solr . . . . .	64
2.5. Patrones base . . . . .	74
3.1. Estados definidos para las anotaciones . . . . .	94
3.2. Vocabularios soportados para la extracción de instancias . . . . .	96
3.3. Lista de configuraciones para el proceso de anotación . . . . .	102
3.4. Información utilizada para la visualización de Servicios Anotados	104
4.1. Palabras entrantes al componente del Servicio Eventful . . . . .	110
4.2. Pruebas mediante Búsqueda directa en Datahub (número reducido de palabras) . . . . .	111
4.3. Taxonomías resultante utilizando AlchemyApi . . . . .	114
4.4. Palabras organizadas por relevancia según AlchemyApi (10 primeras)	115
4.5. Recursos resultantes de la consulta a Datahub (3 primeros) . . . . .	116
4.6. Conceptos más importantes según AlchemyApi . . . . .	118
4.7. Tipos de entidades soportadas por GATE . . . . .	119
4.8. Entidades obtenidas para el servicio Eventful usando GATE . . . . .	120
4.9. Resultados para el servicio Eventful usando AlchemyApi . . . . .	121
4.10. Resultados de la clasificación del Servicio Web Eventful . . . . .	124
4.11. Características de Buscadores semánticos . . . . .	127
4.12. Pruebas Watson . . . . .	129
4.13. Pruebas buscador semántico Falcons . . . . .	133
4.14. Medios de almacenamiento de las ontologías . . . . .	142
4.15. Filtros usados en Solr . . . . .	148
4.16. Petición POST a la primera colección: Concatenación de conceptos	149
4.17. Parámetros del Servicio Web . . . . .	157
4.18. Tabla de métodos de búsqueda implementados con sus números .	158
4.19. Resumen de los métodos para la búsqueda y selección de ontologías	162
5.1. Matriz de confusión . . . . .	165
5.2. Matriz de confusión multiclas . . . . .	166



5.3. Ejemplo de matriz de confusión multiclas . . . . .	167
5.4. Resultados de las fórmulas aplicadas sobre la matriz de confusión de ejemplo . . . . .	167
5.5. Tabla con los resultados del proceso de Anotación Semántica . . . . .	176
A.1. Listado de Servicios Web utilizados durante la evaluación . . . . .	182
A.2. Gold Standard: Resultados de la anotación de los expertos . . . . .	183
A.3. Resultados con ontologías de dominio específico . . . . .	184
A.4. Resultados incluyendo una ontología multidominio . . . . .	184



Yo, *José Enrique Ortiz Vivar*, autor de la tesis *Plataforma para la Anotación Semántica Automática de Servicios Web RESTful sobre un Bus de Servicios*, certifico que todas las ideas, opiniones, y contenidos expuestos en la presente investigación, son de exclusiva responsabilidad de sus autores.

Cuenca, Octubre del 2015.

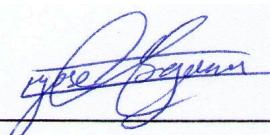
UNIVERSIDAD DE CUENCA  
desde 1867



Yo, *José Luis Segarra Flores*, autor de la tesis *Plataforma para la Anotación Semántica Automática de Servicios Web RESTful sobre un Bus de Servicios*, certifico que todas las ideas, opiniones, y contenidos expuestos en la presente investigación, son de exclusiva responsabilidad de sus autores.

Cuenca, Octubre del 2015.

UNIVERSIDAD DE CUENCA  
desde 1867

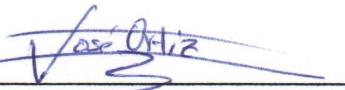


---

José Luis Segarra Flores  
C.I. 0106206642

Yo, *José Enrique Ortiz Vivar*, autor de la tesis *Plataforma para la Anotación Semántica Automática de Servicios Web RESTful sobre un Bus de Servicios*, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de *Ingeniero en Sistemas*. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Cuenca, Octubre del 2015.

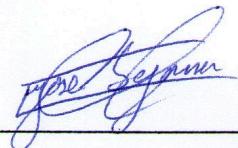


---

José Enrique Ortiz Vivar  
C.I. 0302580683

Yo, *José Luis Segarra Flores*, autor de la tesis *Plataforma para la Anotación Semántica Automática de Servicios Web RESTful sobre un Bus de Servicios*, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de *Ingeniero en Sistemas*. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Cuenca, Octubre del 2015.



José Luis Segarra Flores  
C.I. 0106206642

## CERTIFICO

Que el presente proyecto de tesis: *Plataforma para la Anotación Semántica Automática de Servicios Web RESTful sobre un Bus de Servicios* fue dirigido por mi persona.

UNIVERSIDAD DE CUENCA  
desde 1867

---

**Ing. Víctor Saquicela, PhD.**  
**Director de Tesis**

---

## Agradecimientos

---

A Dios, por permitirnos llegar a la culminación de nuestras carreras universitarias.

A nuestros padres y hermanos, que con su amor y paciencia nos enseñaron a actuar siempre con valor y perseverancia en la vida.

A nuestro director de tesis, Ingeniero Víctor Saquicela, por dedicar su tiempo y conocimientos en la tutoría de este proyecto.

A la dirección de Investigación de la Universidad de Cuenca (DIUC), por colaborar con la presente tesis, enmarcándola dentro del proyecto: “Integración, almacenamiento y explotación de datos hidrometeorológicos utilizando Big Data y Web Semántica”.

A la Facultad de Ingeniería de la Universidad de Cuenca, por darnos la oportunidad de estudiar y ser profesionales.

A nuestros compañeros y amigos, que nos han acompañado a lo largo de nuestra formación profesional.

---

## Dedicatoria

---

A mis padres Delia y Milton, por haber sido los pilares fundamentales de mi formación personal. A mi hermana Gabriela y mi sobrina Sammy, por estar siempre ahí apoyandome en cada paso que doy. A mi incondicional amigo el Rvdo. Padre Galo Gallardo, que con su ejemplo me enseñó que todo es posible si hay fuerza de voluntad. Gracias por confiar en mí.

José Enrique Ortiz Vivar.

UNIVERSIDAD DE CUENCA  
desde 1867

Quiero dedicar esta tesis a mis padres que a pesar del tiempo y las dificultades, nunca han dejado de velar por mí ni un solo día. A mi familia en general, que a pesar de las pocas veces que he tenido la oportunidad de compartir con ellos debido a mis estudios, siempre me han tenido presente. A mis amigos y compañeros que e conocido durante el trayecto de mi vida, muchas de ellos personas talentosas que han servido como marco de referencia constante para superarme a mí mismo y ofrecer mi mejor esfuerzo.

José Luis Segarra Flores.

# **Capítulo 1**

---

## **Introducción**

---

En el presente capítulo se aborda la descripción general del proyecto de tesis, en el que se trata sus antecedentes, problemas actuales, justificación y alcance del proyecto, así como sus objetivos los cuales sirven como soporte para el desarrollo del mismo. Adicionalmente, se describe la estructura utilizada dentro del presente documento, con la cual se busca brindar una rápida orientación en la organización de los temas expuestos.

### **1.1. Antecedentes**

Los Servicios Web en los últimos tiempos han ganado bastante popularidad en muchos de los ámbitos de la Web. Se puede encontrar Servicios Web diseñados para brindar: información de diversos ámbitos (clima, deportes, etc.), soporte a transacciones comerciales y otros cientos de aplicaciones que con el avance tecnológico se suman a esta propuesta. Gran parte del éxito que ha llevado a los Servicios Web a alcanzar dicha popularidad se debe a que poseen muchas de las características deseables en ámbitos heterogéneos como: bajo acoplamiento, procesamiento distribuido e independencia de la plataforma. Lo que les ha brindado una versatilidad notable, que es evidenciada en el gran crecimiento de esta tecnología así como de las aplicaciones que hacen uso de ella. Sin embargo, este acelerado crecimiento que en primera instancia ha favorecido a la expansión y diversificación de Servicios Web, también ha provocado la aparición de algunas dificultades, pues tareas como búsqueda, descubrimiento y composición de servicios frente a esta ingente cantidad de opciones se han vuelto tareas laboriosas realizadas en gran medida de forma manual.

Para remediar el arduo trabajo manual que requieren las actividades mencionadas, se han planteado algunas propuestas que buscan añadir una capa adicional de información a la descripción de Servicios Web, con el fin de lograr cierto grado de automatización sobre las mismas. Esta capa adicional de información agrega una descripción semántica y brinda la posibilidad de convertir a las descripciones sintácticas de los servicios tanto *REpresentational State Transfer* (REST) como *Simple Object Access Protocol* (SOAP) en algo fácilmente entendible por los usuarios y a su vez procesable por las máquinas, dando paso a un nuevo tipo de Servicios Web conocidos como Servicios Web Semánticos.

## 1.2. Planteamiento del problema

La mayoría de propuestas que buscan la adición de semántica a los servicios para la creación de Servicios Web Semánticos, se han enfocado en proponer formalismos como: Micro-WSMO[41], SA-REST[69], SWSAL[61], etc. Lo que a dado paso a la aparición de múltiples lenguajes de descripción semántica que con sus características propias, buscan enriquecer los diferentes elementos que conforman a los Servicios Web. Sin embargo, dichos lenguajes que ofrecen la posibilidad de generar Servicios Semánticos no han profundizado en la definición de procedimientos que permitan llevarlos a cabo. Esto ha provocado que los desarrolladores que requieren la adopción de estas propuestas, terminen implementándolas de forma manual. Esta necesidad de esfuerzo adicional, ha incitado que los enfoques de descripción semántica tengan poca aceptación fuera de entornos de investigación y por lo tanto la adopción de Servicios Web Semánticos en la Web se vea limitada.

Por otro lado, el reducido número de propuestas enfocadas en superar la falta de automatización en la adopción de estos formalismos, aún se encuentran en investigación y depende todavía de un importante nivel de intervención por parte del usuario.

## 1.3. Justificación

Actualmente como uno de los esfuerzos para la adopción de Servicios Web Semánticos, se han generado algunas propuestas dentro del estado del arte con los que se busca facilitar el proceso de anotación de Servicios Web sintácticos para convertirlos en semánticos. Dentro de estas propuestas se pueden encon-

trar a *Semantic Web Services Editing Tool* (SWEET), que provee una táctica de anotación asistida para el usuario, que busca simplificar y disminuir la carga de tiempo y conocimiento necesario para realizar anotaciones sobre los Servicios REST. Dicho enfoque aunque representa un importante avance respecto a la anotación automática de este tipo de servicios, aún requiere de un gran porcentaje de intervención manual y de validación por parte de expertos haciendo difícil su adopción. Por otro lado, un enfoque más dirigido hacia la automatización verdadera de las anotaciones semánticas sobre Servicios Web se puede encontrar en la propuesta de Saquicela y otros [67] [21]. Esta propuesta tiene un planteamiento conformado por varias etapas que con la ayuda de servicios externos busca dotar de un nivel de automatización casi completo del proceso de anotación de servicios. Sin embargo, este planteamiento aún omite algunos pasos, como la etapa de selección de ontologías con la cual realizar el enlace de conceptos, ya que regularmente dicha selección depende del dominio del servicio. Por esta razón y con el fin de brindar una plataforma enfocada hacia la anotación de Servicios Web REST de manera automática se ha planteado durante la presente tesis:

- La adopción del enfoque de anotación planteado en [67] [21] como base de la plataforma.
- El planteamiento e implementación de una etapa de selección de ontologías de manera automática para dotar al enfoque de anotación semántica adoptado, de un nivel más alto de automatización.
- El desarrollo de un prototipo que contenga el planteamiento adoptado y mejorado de anotación semántica sobre un Bus de Servicios Empresarial, con el que se facilite comprobar el funcionamiento de la propuesta con cualquier cantidad y ámbito de servicios.
- La evaluación del proceso de anotación automática de Servicios Web usando servicios de varios dominios.

Con la presente propuesta se pretende aportar al estado del arte en el ámbito de automatización de anotaciones semánticas de Servicios Web REST y colaborar de esta manera para que la implementación de Servicios Web Semánticos con sus múltiples beneficios, llegue a ser adoptada de manera más extendida a través de la Web.

## 1.4. Objetivos

En la siguiente sección se describe los objetivos tanto generales como específicos que se pretenden alcanzar con el presente proyecto.

### 1.4.1. Objetivo general

El aporte propuesto en la presente tesis tiene como objetivo: la implementación de una plataforma prototipo de Anotación Semántica de Servicios Web REST sobre un Bus de Servicios que genere anotaciones semánticas de manera automática.

### 1.4.2. Objetivos específicos

- Analisar el estado del arte de los distintos enfoques para Anotación Semántica de Servicios Web.
- Investigar sobre distintas arquitecturas orientadas hacia servicios, patrones de integración y sobre las diferentes implementaciones requeridas para el manejo, integración y anotación de servicios.
- Implementar y extender el enfoque de Anotación Semántica de Servicios Web mediante un proceso de selección automática de ontologías.
- Probar y evaluar el proceso de Anotación Semántica de Servicios Web usando la selección automática de ontologías.

## 1.5. Alcance

El presente proyecto de tesis pretende el diseño y desarrollo de una plataforma prototipo, para la anotación semántica de Servicios Web REST de forma automática, con la ayuda de una etapa de selección automática de ontologías. Para lo cual en primer lugar se investigará tecnologías existentes que se orienten a la automatización del proceso de anotación semántica, tomando como referencia la tesis doctoral presentada en [21].

Posteriormente se adoptará y expandirá el planteamiento seleccionado, incluyendo una etapa de selección automática de ontologías que permita aumentar

el nivel de automatización del proceso, así como el ámbito de dominios de los Servicios Web que se puedan anotar.

Finalmente se procederá a la implementación del prototipo utilizando para esto tecnologías enfocadas en el manejo de Servicios Web e integración, lo que permitirá la realización de pruebas sobre los distintos métodos de selección de ontologías planteados, así como la extracción de resultados de la ejecución completa del proceso de anotación semántica automática de Servicios Web, para un conjunto de servicios de prueba.

## 1.6. Estructura del documento

La organización de la tesis está conformada de la siguiente manera. En el capítulo 2, se hace una descripción del marco teórico respecto a las tecnologías usadas, así como del estado del arte respecto a la anotación semántica de Servicios Web REST. En el capítulo 3, se describe el análisis, modelamiento e implementación de la plataforma de anotación semántica propuesta. En el capítulo 4, se detalla creación del componente de selección de ontologías. En el capítulo 5, se expone el proceso de evaluación y resultados a los que se ha llegado durante el desarrollo de este trabajo. En el capítulo 6, se indican finalmente, las principales conclusiones obtenidas con la presente tesis, así como trabajos futuros que pueden llegar a realizarse a partir de esta propuesta.

## **Capítulo 2**

---

### **Marco teórico**

---

En este capítulo se abordan las principales tecnologías y conceptos ligados al desarrollo de la presente tesis, temas que permiten tener la base teórica necesaria para lograr la comprensión del planteamiento propuesto y realizar el seguimiento adecuado de su desarrollo.

Entre los principales temas presentados a continuación se encuentran : Servicio Web tradicionales, Tecnologías de la Web Semántica, Anotación Semántica de Servicios Web, Tecnologías de Integración (Buses de Servicios Empresarial).

#### **2.1. Servicios Web**

Los Servicios Web son una tecnología para el intercambio de información y se han vuelto muy populares en la actualidad. Grandes compañías como Google<sup>1</sup>, Yahoo<sup>2</sup>, Amazon<sup>3</sup> y una gran cantidad de otras empresas hacen uso extendido de esta tecnología. Se pueden encontrar por ejemplo la aplicación de estos servicios en la publicación de información de reportes de clima de un determinado lugar, la recepción y envío de noticias de todo el mundo y la compra-venta de productos en línea. Sin embargo, dichos ejemplos no son más que una pequeña muestra del éxito creciente que ha llegado a tener este tipo de servicios sobre muchos ámbitos de la Web. Gran parte del éxito de estos servicios, se debe al aprovechamiento del gran potencial que tiene la Web, entre los que se encuentran la capacidad de conexión entre las redes, disponibilidad de procesamiento distribuido y la capacidad de intercambio de información independiente del lugar y el tiempo, entre muchos otros.

---

<sup>1</sup>[www.google.com](http://www.google.com)

<sup>2</sup>[www.yahoo.com](http://www.yahoo.com)

<sup>3</sup>[www.amazon.com](http://www.amazon.com)

Frente a este escenario, muchos proveedores de servicios han diseñado e implementado un gran abanico de Servicios Web que además de ser reutilizables y específicos para determinadas tareas, evitan el desarrollo por parte del cliente de la misma funcionalidad. Otra de las razones por la que los Servicios Web han llegado a ser una tecnología muy apreciada, es por su flexibilidad frente a diversas plataformas heterogéneas, algo que es indispensable en la Web. Por lo que se puede acceder de la misma forma desde una computadora, con Windows, Linux, Mac e incluso desde móviles con sistemas como Android, Windows Phone, etc. mediante el uso de protocolos y tecnologías estándar. La adopción de estos servicios además ha evitado la necesidad de emplear un middleware específico como muchos de los otros sistemas distribuidos (RPC<sup>4</sup>, CORBA<sup>5</sup>, DCOM<sup>6</sup>, etc.) y al mismo tiempo ha logrado conseguir funcionalidades similares que satisfacen un conjunto de necesidades latentes dentro de Internet [31].

Pero formalmente ¿Qué es un Servicio Web?. No existe una definición estándar de lo que es un Servicio Web. Por lo tanto la definición de un Servicio Web puede llegar a ser más genérica o más específica según la perspectiva de la que se plantee. Sin embargo la definición que provee la W3C<sup>7</sup> es bastante aceptada [32], en el cual un Servicio Web es “una aplicación de software identificada por una URI, cuyas interfaces y enlaces son capaces de ser definidos, descritos y descubiertos como artefactos *Extensive Markup Language* (XML). Un servicio Web soporta interacción directa con otros agentes de software que usan mensajes basados en XML que son intercambiados por protocolos basados en Internet”[18]. Aunque esta definición está más enfocada a un tipo de servicios (SOAP) se puede presentar una idea más general de los servicios para abarcar otros tipos como REST mediante la generalización de esta definición. Por lo que se puede afirmar que los Servicios Web son aplicaciones accesibles por medio de Internet y que utilizan tecnologías estándar para poder comunicarse y proveer su funcionalidad. Una de las características de los Servicios Web en general es que poseen bajo acoplamiento y encapsulamiento, lo que significa que se puede llegar a utilizar independientemente de cómo este desarrollado el servicio, además de no ser accesible a la implementación por quién lo utiliza. Esto simplifica su uso debido a que el cliente únicamente debe preocuparse por cómo acceder al servicio y cómo debe enviar y recibir los mensajes en lugar de cómo se realiza la operación que solicita.

---

<sup>4</sup><http://www.cs.cf.ac.uk/Dave/C/node33.html>

<sup>5</sup><http://www.corba.org/>

<sup>6</sup><http://opengroup.org/comsource/>

<sup>7</sup>[www.w3.org](http://www.w3.org)

Existen diferentes tipos de Servicios Web pero se tratarán dos específicamente, que son de interés para la presente tesis. Estos son los Servicios Web REST y los servicios SOAP.

### 2.1.1. Servicios SOAP

Los Servicios Web SOAP son servicios que utilizan el protocolo de comunicación SOAP y otras tecnologías relacionadas como *Web Service Description Language* (WSDL), *Universal Description, Discovery and Integration* (UDDI) para su funcionamiento, como se muestra en la figura 2.1. SOAP es un protocolo que permite utilizar XML para transferir información entre nodos (solicitante y receptor) y nace como una recomendación de la W3C después de un amplio análisis y consenso para crear un protocolo liviano, independiente de la plataforma, independiente del transporte e independiente del sistema operativo. Características que son necesarias para el trabajo sobre ambientes heterogéneos requeridos por los Servicios Web [31] y que se logran gracias al uso de tecnologías estándar (XML, *Hypertext Transfer Protocol* (HTTP)). Las siglas SOAP significan Simple Object Access Protocol pero desde la versión 1.2<sup>8</sup> se le denomina solamente SOAP debido a que ya no trabaja directamente con objetos [76]. La comunicación de SOAP fundamentalmente es sin estados y posee un paradigma para intercambio de información de una vía, pero se puede crear patrones más complejos por ejemplo petición-respuesta o una petición-múltiples respuestas [60]. También se tiene variaciones en las que se simula un estado para atender demandas de autenticación o sesión.

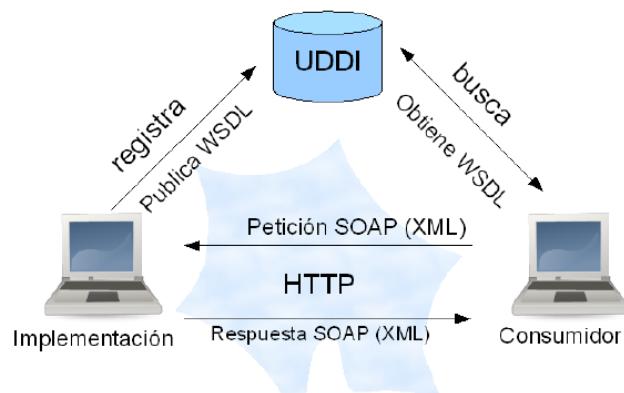


Figura 2.1: Elementos de un WS SOAP [23]

<sup>8</sup><http://www.w3.org/TR/2001/WD-soap12-20010709/>

### 2.1.1.1. Mensajes SOAP

Los mensajes en SOAP no son más que documentos estandarizados en XML que consta de los siguientes elementos, englobados por el elemento raíz *envelope*:

**Header:** Contiene información adicional sobre el mensaje, por lo que es opcional. Este puede contener información por ejemplo de autentificación o de codificación de los datos. Adicionalmente puede contener mensajes para los intermediarios<sup>9</sup>.

**Body:** Contiene el mecanismo para el intercambio de información con el destinatario final del mensaje. Puede contener un documento XML o una llamada RPC<sup>10</sup> con datos estructurados e informe de fallos.

Adicionalmente al protocolo SOAP, los Servicios Web requieren formas que permita descubrir al Servicio Web por parte de los consumidores y una forma para que estos sepan cómo comunicarse con el servicio. Para eso se utiliza las tecnologías WSDL y UDDI.

### 2.1.1.2. WSDL

Lenguaje de Descripción de Servicios (WSDL) es un archivo XML que provee información técnica sobre cómo esta implementado un servicio. Está compuesto por *Uniform Resource Identifier* (URI), nombres de los métodos, argumentos y tipos de datos [76]. Esto permite al solicitante del servicio crear su propio mensaje SOAP para comunicarse con el proveedor y acceder al servicio.

La característica destacada que brinda WSDL, respecto a un descripción no estandarizada, como una página Web de información, es que puede ser usado por las máquinas para automatizar el proceso de creación de mensajes SOAP. Sin embargo, al ser únicamente una descripción sintáctica del servicio, no es capaz de relacionar sus elementos con otros, ni de reconocerlos de otra forma que no sea para la generación de la llamada al servicio.

Una vez se tiene el WSDL, es necesario publicarlo para que pueda ser accesible y utilizable por los usuarios; con este fin fue desarrollado UDDI el cual se describe a continuación.

---

<sup>9</sup>Pueden existir intermediarios en el mensaje como proxys, routers entre otros que también pueden ver información, pero únicamente la especificada en el encabezado.

<sup>10</sup>Remote call Procedure : Se expresa la información como si se tratase de un función local pero es descrita en formato XML.

### 2.1.1.3. UDDI

Descripción Universal de Descubrimiento e Integración (UDDI) es un directorio estándar donde se pueden publicar los Servicios Web, y en los cuales consta la información necesaria para que los posibles usuarios puedan acceder al servicio. UDDI sirve de intermediario para localizar los servicios requeridos por parte de los solicitantes del servicio hacia los proveedores del servicio [5]. Los datos pueden llegar a organizarse de la siguiente manera:

**Información de negocio:** Permite encontrar a la empresa que publicó el servicio dentro de un ámbito de la industria o producto.

**Información de enlace:** Información técnica del servicio y de cómo comunicarse con él.

**Información del proveedor del servicio:** Información sobre quién publica el servicio, incluyendo dirección, contacto, etc.

Aunque UDDI provee una alternativa para el descubrimiento de servicios, actualmente no es muy utilizado debido a que la búsqueda se centra únicamente en información sintáctica de los metadatos, obviando por ejemplo la descripción del servicio [70].

### 2.1.1.4. Ejemplo mensaje SOAP

Para realizar una petición a un servicio SOAP, la información debe estructurarse en XML, con los elementos anteriormente mencionados. En este caso, el segmento de código 2.1, muestra la petición de la información de un servicio por parte de un cliente que requiere la información de determinado producto. Esto se realiza mediante una llamada a la función **getProductDetails**, pasando como parámetro el id del producto. La respuesta a este mensaje por parte del proveedor sería otro mensaje con SOAP, como se muestra en la segmento de código 2.2.

```
1<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/envelope/">\n2  <soap:Body>\n3      <getProductDetails xmlns="http://ware.example.com/ws">\n4          <productId>827635</productId>\n5      </getProductDetails>\n6  </soap:Body>\n7</soap:Envelope>
```

Segmento de Código 2.1: Ejemplo solicitud de servicio usando mensaje SOAP

```
1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/envelope/">
2   <soap:Body>
3     <getProductDetailsResponse xmlns="http://ware.example.com/ws">
4       <getProductDetailsResult>
5         <productName>Toptimate 3-Piece Set</productName>
6         <productId>827635</productId>
7         <description>3-Piece luggage set. Black Polyester.</description>
8         <price>96.50</price>
9         <inStock>true</inStock>
10        </getProductDetailsResult>
11      </getProductDetailsResponse>
12    </soap:Body>
13  </soap:Envelope>
```

Segmento de Código 2.2: Ejemplo de respuesta SOAP

### 2.1.2. Servicios REST

Los Servicios Web REST son una propuesta de arquitectura de software para sistemas hipermedia distribuidos, tales como la Web [58]; estos servicios han venido ganando popularidad actualmente, superando incluso a servicios como SOAP. El término REST fue introducido por Roy Fielding<sup>11</sup> en su tesis doctoral, al cual también se le atribuye la creación del protocolo HTTP. Por si solo los Servicios Web REST no son un estándar, pero se basan en algunos estándares como HTTP y *Uniform Resource Locator* (URL). En el caso de URL más específicamente utiliza URI para la identificación de sus recursos. Para la representación de resultados también utiliza estándares como XML, *HyperText Markup Language* (HTML), JPEG, entre otros.

Los Servicios Web REST tienen las siguientes características según [5]:

1. Uso de interfaces generales para manipular recursos por medio de HTTP.

- Get: Obtener la representación de un recurso.
- Delete: Se usa para eliminar representaciones de un recurso.
- Post: Usado para actualizar o crear las representaciones de un recurso.
- Put: Se usa para crear representaciones de un recurso.

<sup>11</sup><http://www.ics.uci.edu/~fielding/>

2. La mayoría de mensajes deben ser XML.
3. Los mensajes deben ser simples y son codificados en las URLs.
4. Los servicios y los proveedores deben ser recursos, mientras que el consumidor puede ser un recurso.

### 2.1.2.1. Descripción del servicio REST

La información de descripción de los servicios REST a diferencia de los servicios SOAP, suele encontrarse en una página Web de información. Dicha página Web suele contener la información acerca de los funcionalidades del servicio, así como los parámetros de entrada y salida. La descripción del servicio al ser netamente sintáctica, requiere la intervención de un usuario o programador para su implementación de forma que pueda ser formada la llamada al servicio, así como el tratamiento de los datos de salida.

### 2.1.2.2. Ejemplo de llamada a un servicio REST

Los Servicios Web REST son considerados bastante simples de acceder, debido a que la petición que se hace al proveedor del servicio se la realiza normalmente utilizando parámetros dentro de la misma URL de invocación. La URL que se utilizará de ejemplo se encuentra en 2.3 y pertenece a un servicio disponible en la página Traileraddict<sup>12</sup>. Este servicio al igual que la función de su página principal es la de presentar avances de películas a los usuarios. En este ejemplo se indica que se requiere trailers de películas con el nombre Titanic. La respuesta en formato XML se puede ver en el espacio de código 2.4.

<sup>1</sup> <http://api.traileraddict.com/?film=Titanic>

Segmento de Código 2.3: URL de un servicio REST

```
1<trailers>
2<trailer>
3<title>Titanic: Belfast Museum – Newswrap</title>
4<link>
5  http://www.traileraddict.com/titanic/belfast-museum-newswrap
6</link>
7<pubDate>Sun, 23 Sep 2012 07:04:25 -0700</pubDate>
8<trailer_id>62863</trailer_id>
9<imdb>0120338</imdb>
```

<sup>12</sup><http://www.traileraddict.com/>

```

10<embed>
11<! [CDATA[
12<iframe width="480" height="270" src="//v.traileraddict.com/62863"
    allowfullscreen="true" Webkitallowfullscreen="true"
    mozallowfullscreen="true" scrolling="no" frameborder="0"></iframe>
13<p><a href="http://www.traileraddict.com/titanic/belfast-museum
    -newswrap">Belfast Museum – Newswrap</a> for <a href="http://www.
    traileraddict.com/titanic">Titanic</a> on <a href="http://www.
    traileraddict.com">TrailerAddict</a>.</p>
14]]>
15</embed>
16</trailer>
17</trailers>

```

Segmento de Código 2.4: Respuesta del servicio

### 2.1.3. Servicios Web REST o Servicios Web SOAP

Frente a la aparición masiva de los Servicios Web tanto de arquitectura REST como SOAP, es natural entre muchas de las consideraciones ha tomarse en cuenta, el elegir entre uno de estos tipos de servicios. Con el afán de concentrar los esfuerzos de la presente tesis sobre un tipo de servicio específico, se ha realizado una tabla comparativa de las características relevantes que posee cada uno (Tabla 2.1), lo cual ha facilitado la selección del tipo de servicio más apropiado, tomando como consideraciones relevantes su impacto en la actualidad y el futuro.

Características	REST	SOAP
Usabilidad	Fácil de usar, pues depende únicamente de parámetros.	Requiere archivos complejos (WSDL)
Automatización	Difícil de automatizar	Puede configurarse automáticamente la comunicación con ayuda del descriptor del servicio (WSDL)
Aplicabilidad	Usado preferiblemente en Internet	Popular en ambientes locales.
Ámbito de aplicación	Usado en múltiples ámbitos	Usado en ámbitos que requieren mayor seguridad y fiabilidad.

Tabla 2.1: Comparativa REST vs SOAP

Como se puede apreciar en la tabla comparativa entre Servicios Web REST y SOAP, los servicios REST son los servicios que han ganado mayor presencia en la Web, debido principalmente a su capacidad de funcionar en ambientes heterogéneos con bajo acoplamiento y facilidad de uso. Hecho que se puede co-

rroborar con los resultados presentados por Google Trend<sup>13</sup>, en donde se analizan las búsquedas dominantes en Internet como se muestran en la figura 2.2, donde es evidente el crecimiento de REST respecto a SOAP. Otra forma en que se ha analizado el uso y disponibilidad sobre los diferentes tipos de servicios es mediante las estadísticas del sitio de alojamiento de API de Servicios Web programmableWeb<sup>14</sup>. En la figura 2.3 se muestra algunas estadísticas de los servicios que alberga dicha página, en donde se puede notar claramente que los servicios de tipo REST dominan en número a los demás servicios, quedando en segundo lugar los servicios SOAP. Por lo tanto siguiendo la tendencia donde los servicios REST son los que poseen mayor popularidad, y por otro lado los que menor capacidad tienen de automatización, estos han sido seleccionados para ser tratados en la presente tesis como objetos de prueba, en la cual se buscará su enriquecimiento mediante anotaciones semánticas para operaciones como búsquedas, composición y descripción de servicios puedan ser automatizables.

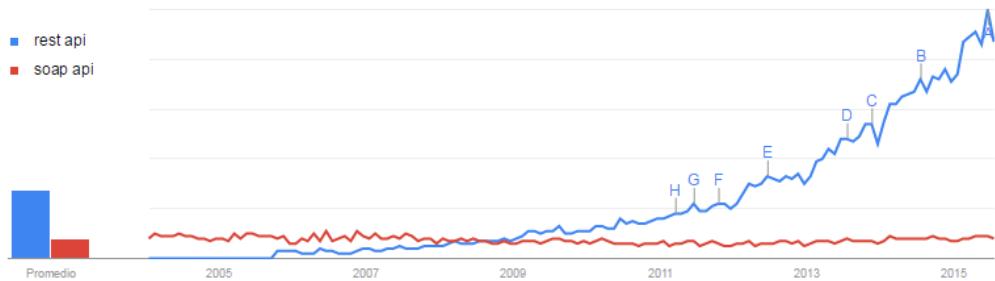


Figura 2.2: Nivel de interés de búsquedas de servicios en el tiempo

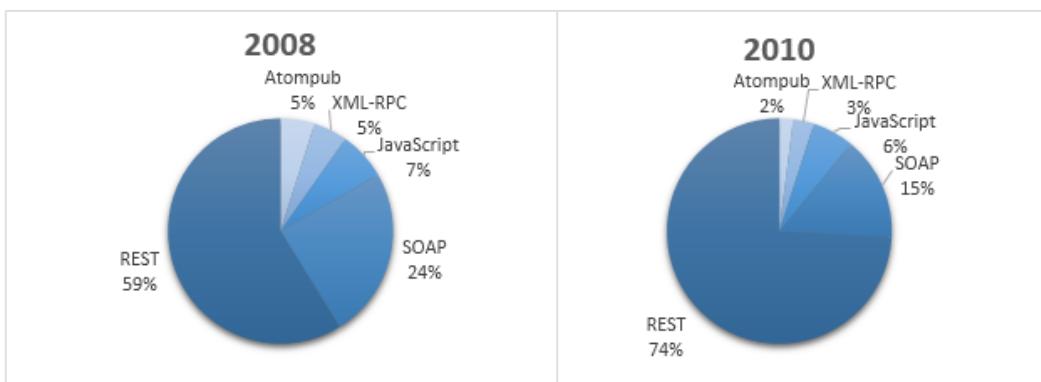


Figura 2.3: Porcentaje de Servicios Web contenidos en programmableWeb

<sup>13</sup><http://www.google.es/trends/>

<sup>14</sup><http://www.programmableWeb.com/>

### 2.1.4. Composición de Servicios

Una de las operaciones relacionadas con los Servicios Web que han ido tomando presencia principalmente con la aparición de los servicios semánticos, es la composición de servicios. Esta operación surge de la necesidad de encontrar una funcionalidad determinada a través de un servicio, para el cual ningún servicio desarrollado la cumple. Por lo que para suplir dicha falta se recurre a la creación de un nuevo servicio a partir de los servicios existentes [24]. Dependiendo del tipo de servicios y el ámbito en que se forma dicha composición suelen referirse a los mismos como “Mashup” para los servicios de tipo REST y orquestación para los servicios SOAP. Dado que este tema no es fundamental para los propósitos de la presente tesis no se profundizara en el tema, sin embargo aquellos interesados en conocer acerca de los mecanismos de composición de servicios usando servicios REST pueden consultar [7], mientras que aquellos interesados en orquestación de servicios SOAP podrán encontrar mayor información en [64].

## 2.2. Web Semántica

En la presente sección se describe la evolución que ha tenido la Web, hasta llegar a la aparición de lo que hoy se conoce como Web Semántica, así como las principales características de esta tecnología que han permitido destacar a la misma de cara al futuro.

### 2.2.1. Antecedentes

La Web como se ha venido manejando aunque ha logrado mucho éxito alrededor del mundo debido a las múltiples ventajas que presenta, como la facilidad de acceso a recursos de información, reducción de distancias a través de la interconexión de la red y aplicabilidad en diferentes áreas de interés como salud, educación y negocios, se ha quedado corta frente a las necesidades de esta nueva era. Este tiempo en donde la información y el conocimiento han cobrado principal protagonismo, tanto dentro de negocios como en los avances tecnológicos y científicos, es imperativo el incluir a la Web dentro de este ámbito, en busca de la Web del conocimiento [34]. Por esta razón, personas como Tim Berners-Lee han presentado propuestas como Linkend Data y Web semántica para que la extracción de conocimiento a partir de esta fuente enorme de información como es el Internet y la Web en general sea posible. Dichas propuestas sin embargo, para

ser aplicables aún tienen que superar muchos problemas de la Web actual, que parten del enfoque algo limitado que se ha dado inicialmente a la Web. Al principio la Web fue pensada como un medio para la presentación de la información a los usuarios (Web 1.0), pero luego dada la necesidad que los usuarios tenían de interactuar y aportar la información a la misma, evolucionó convirtiéndose en una Web dinámica (Web 2.0). Sin embargo hasta hace poco, aún no se había considerado la necesidad de aprovechar la información recopilada para darle un sentido explícito que sea apreciado tanto por las máquinas como por las personas, que es lo que pretende conseguir la Web Semántica (Web 3.0) [68].

### 2.2.2. Definición de Web Semántica

La Web Semántica es definida como una extensión de la Web actual, en la cual la información tiene un significado bien definido, permitiendo que el trabajo conjunto entre las personas y computadoras sea posible de mejor manera. La idea central de la Web Semántica se basa en que las computadoras no sean utilizadas únicamente como máquinas que presentan información, sino que por medio de las computadoras se logre relacionar la información incluso de distintas fuentes, reconociendo si la información es similar o no, incluso infiriendo nueva información si es que esta no se encuentra explícita. El objetivo de esta nueva Web por lo tanto es crear un medio universal para intercambio de datos, lo cual permitirá que datos de uso personal, de gobiernos, compañías sean fácilmente automatizados, integrados, y utilizados [77].

Se puede apreciar la necesidad de la Web Semántica claramente con un ejemplo cotidiano como la búsqueda en la Web. Para acceder al contenido que se encuentra en esta, el paso más común es utilizar la ayuda de un buscador. Los buscadores para proporcionar la información requieren el uso de palabras claves que representen los resultados que se desean obtener. Estas palabras están indexadas para agilizar su búsqueda, y lo relacionan con los documentos en las que se encuentran, sin embargo, esto ocasiona que los resultados dependan en gran medida de las palabras que se ingresan y la cantidad de datos que se obtienen sea grande aún cuando realmente pocos son los resultados que reflejan lo que se buscaba inicialmente. La falta de sentido a la información por parte de las máquinas que sirven de intermediarios entre los usuarios y las búsquedas son la principal causa para este gran limitación. Aún cuando se han realizado grandes esfuerzos por enfrentar y en parte mejorar los resultados de las búsquedas con el uso de algoritmos mejorados, reconocimiento de lenguaje natural e inteligencia

artificial, como lo demuestra [37], no se ha podido crear soluciones generales, debido a que hace falta una estandarización que permita dar sentido verdadero a la información [3].

Para la realización de la Web Semántica se necesita un conjunto de tecnologías y un marco de trabajo que los soporte, entre los cuales se encuentran [40]:

**Contenido semántico:** Se refiere a la combinación entre datos y metadatos.

**Datos:** En ésta se incluyen tanto los datos estructurados como los no estructurados. Por ejemplo XML, *Resource Description Language* (RDF), texto plano.

**Metadatos:** Son los que aportan las anotaciones que los relacionan con la Web Semántica.

**Ontologías y esquemas:** Son colecciones de conceptos que regularmente están relacionadas bajo un dominio. Son las que proveen de vocabulario y sentido semántico a los metadatos.

### 2.2.3. Ontologías

Para la realización de la Web Semántica como se explicó en el sección 2.2.2, son necesarios diversos elementos, donde uno de los centrales son las ontologías. Existe muchos significados respecto a la definición de una ontología, dentro de la literatura pero en el caso de la Web Semántica y la inteligencia artificial que es el contexto en el que se está exponiendo este trabajo, se suele considerar la definición dada por Gruber. Este autor define a la ontología como “una especificación explícita y formal sobre una conceptualización compartida” [30]. Es decir que las ontologías son la especificación de un conjunto de conceptos y relaciones de forma consensuada y compartida acerca de un dominio, la cual es representada de forma legible, formal y utilizable por los computadores [79]. Ésta consta de un conjunto de componentes según lo indica Tello en [79] que son utilizados para representar el conocimiento.

**Conceptos:** Es la base de lo que se pretende formalizar. Pueden ser clases de objetos, métodos, incluso procesos de estrategias, entre otros.

**Relaciones:** Representa el enlace que existen entre los conceptos y su interacción. Por ejemplo Subclase-de, conectado-a, es-igual-a, etc.

**Instancias:** Son objetos determinados de una clase.

**Axiomas:** Teoremas o reglas que se consideran sobre las relaciones que debe cumplirse.

Los principales componentes que son las clases (Class), instancias (Individuals) y propiedades (Relations) [44] son categorizados en dos grupos: TBOX (Terminology o Taxonomy component) y ABOX (Assertion Component). El primer grupo se refiere al esquema o vocabulario de la ontología, es decir las Clases y Propiedades de la ontología que forman la estructura de los datos. El segundo grupo corresponde a las instancias o datos pertenecientes a dichas clases o propiedades.

Para tener una idea más clara de a qué se refiere una ontología, se puede ejemplificar acerca de los elementos que posee una universidad y como están organizados. Una universidad está conformada por personas, las cuales se clasifican en personal y estudiantes. Los estudiantes pueden ser de pregrado o postgrado, mientras que el personal puede ser de soporte, administración o académicos. Esta representación es una jerarquía con elementos que se relacionan los unos con los otros como se puede observar en la figura 2.4. En este caso la relación existente entre los elementos se puede expresar como subclase de (*subClassOf*), indicando que la clase posterior es una subclase de la superior. Aplicando reglas de inferencia que es el mecanismo para el descubrimiento de relaciones no aplicadas explícitamente podemos conocer por ejemplo que un académico (profesor) es parte de las personas en la universidad, también por ejemplo que el personal de administración no consta como estudiante debido a que no existe una relación de subclase con la clase estudiante, aunque si es considerado parte de la universidad.

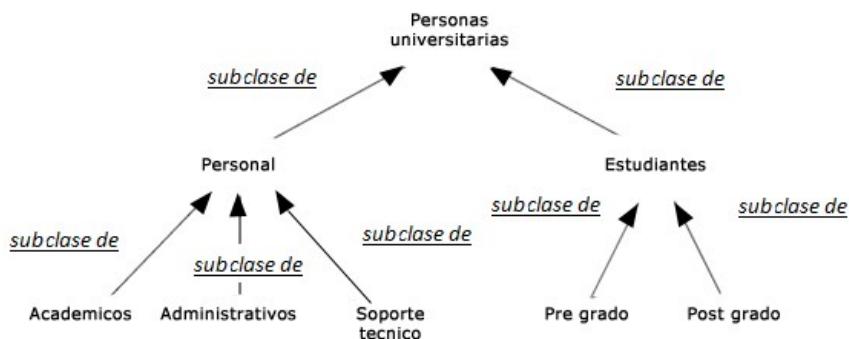


Figura 2.4: Ejemplo de representación por grafos de una ontología

## 2.2.4. Lenguajes ontológicos de la Web Semántica

Para poder representar las ontologías adecuadamente de forma que cumpla con las características mencionadas en 2.2.3, es decir que sean consensuadas, compartidas y compatibles para el manejo por las computadoras, se requiere de un tipo de lenguaje estandarizado que lo soporte. Se han creado varios estándares hasta la fecha dependiendo de las necesidades que han ido surgiendo y la necesidad de expresividad requerida. Entre estos los más destacados y conocidos se encuentran RDF, RDF Schema, *Ontology Web Language* (OWL).

### 2.2.4.1. RDF

Marco de descripción de recursos o RDF es a menudo considerado como un “lenguaje” para la representación de metadatos sobre recursos en la Web [40] aunque, se considera esencialmente como un modelo de datos. En sí, es una propuesta realizada por la W3C<sup>15</sup> en la cual constan un conjunto de recomendaciones que lo describen [29]. La idea central de usar RDF es presentar información semántica usando sentencias conocidas como *statements* o declaraciones, en las cuales se describen a objetos o recursos relacionados con determinadas propiedades a las que se les asigna un valor. Este modelo de Recurso-Propiedad-Valor se le conoce como tripleta, ya que requiere de estos tres elementos (figura 2.5) para formar una declaración [17]. Por lo general RDF ocupa la sintaxis XML para estructurar su contenido lo que se conoce como RDF/XML, aunque existen otras sintaxis de representación como Turtle<sup>16</sup>.

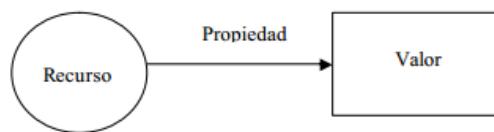


Figura 2.5: Representación de una tripleta [80]

Para la generación de RDF y en general para la descripción de información semántica, se requiere que los elementos puedan ser reconocidos por las máquinas de forma clara y exacta. De otra manera, pudiera presentarse ambigüedad sobre sus elementos (recursos) y relaciones (propiedades). Una de las formas más prácticas para poder expresar claramente que se está tratando con un determinado recurso o elemento específico es hacer uso de un identificador. Las URI

<sup>15</sup><http://www.w3c.es/>

<sup>16</sup>Sintaxis textual para representar tripletas RDF de forma simplificada y compacta

o Universal Resource Identifier es una descripción que se usa para identificar un recurso de manera única en la Web. Un ejemplo de una URI puede ser <http://www.ucuenca.edu.ec/facuingenieria>, que puede ser usado para representar como un recurso a la Facultad de Ingeniería de la Universidad de Cuenca.

Para un mayor entendimiento de lo qué es un recurso y los demás conceptos de la sintaxis en los RDF[3], se describirá a continuación.

**Recursos:** Se refiere a un objeto o cosa sobre la que se está describiendo. Por ejemplo puede ser un libro, un autor, un alumno, un modelo de auto, etc.

**Propiedad:** Son un tipo especial de recursos por lo que pueden ser identificados con URIs. Estos hacen referencia a las relaciones. Por ejemplo, `es igual_a`, `pertenece_a`, etc.

**Declaración:** Afirma una propiedad que posee un recurso. Por ejemplo Juan posee el correo `juanito@correo.com`

Se puede observar un ejemplo de los elementos mencionados anteriormente en la figura 2.6, donde se muestra al recurso <http://ucuenca.ec/contact#JP> relacionado con diferentes propiedades. Dicho ejemplo representa un contacto, el cual posee un número de teléfono, correo, un nombre completo y un título. Se puede destacar del ejemplo el caso del item “*mail*”, donde el valor con el que se relaciona puede ser otro recurso, mientras si es un valor terminal se les denomina individuales, como “*titleperson*” (título de persona), “*fullname*” (nombre completo) o “*phone*” (teléfono). Su representación RDF/XML se puede ver en 2.5.

```

1<?xml version="1.0"?>
2<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
3  xmlns:contact="http://www.ucuenca.ec/contact#">
4<contact:Person rdf:about="http://ucuenca.ec#JP">
5<contact:fullname> Jorge Perez </contact:fullName>
6<contact:mail rdf:resource="mailto: vo@uc.mail"/>
7<contact:phone>545564</contact:phone>
8<contact:titleperson>Jorge Perez</contact:titleperson>
9</contact:Person>
10</rdf:RDF>
```

Segmento de Código 2.5: Ejemplo RDF/XML

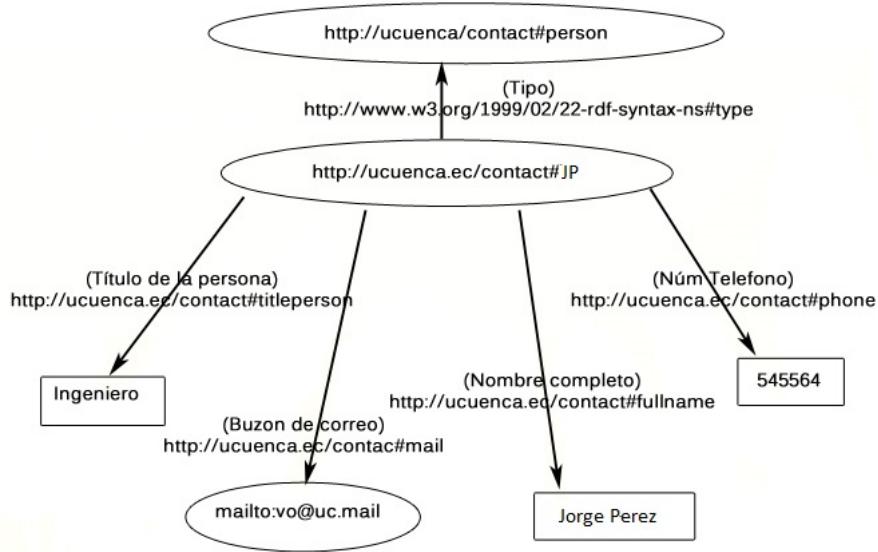


Figura 2.6: Ejemplificación de un grafo RDF

#### 2.2.4.2. RDF Schema

Como se vio en la sección 2.2.4.1, el lenguaje RDF es usado cuando se requiere expresar declaraciones simples acerca de recursos por medio de propiedades y valores. Sin embargo con el uso de este tipo de tecnología, la necesidad de expresar relaciones más complejas como restricciones o de incluir vocabularios concertados por la comunidad fueron haciéndose más evidentes [40]. Es por eso, que para cumplir con nuevos requerimientos se han creado un conjunto de primitivas adicionales utilizables sobre el “lenguaje” RDF para formar uno nuevo, conocido como RDF Schema (RDFS) el cual permite entre sus principales características [12]:

- Definir restricciones de dominio y rango para las propiedades.
- Utilizar vocabularios específicos para expresar clases, instancias de esas clases y jerarquización.

Un ejemplo de un grafo que usa RDF Schema se puede observar en la figura 2.7, donde se puede notar como RDF Schema complementa con un nivel adicional de información a RDF. Este nivel de información adicional por ejemplo permite declarar algunas consideraciones específicas sobre el grafo como que: El personal y los estudiantes son una subclase de (`subClassOf`) personas universitarias.

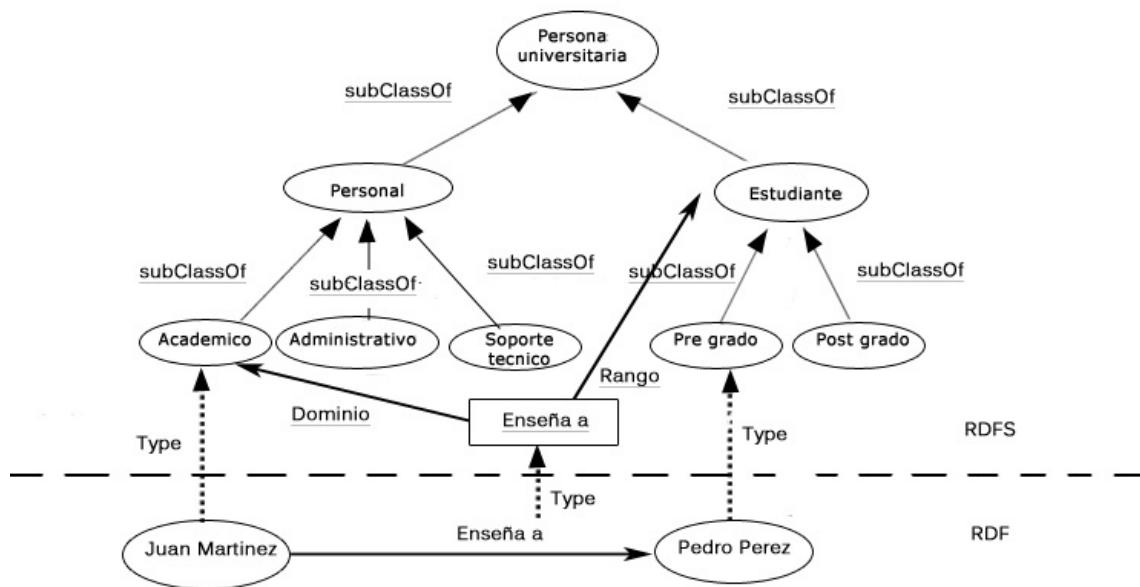


Figura 2.7: Representación con grafos de un ejemplo en RDFS

También se puede encontrar por ejemplo como un recurso u objeto puede llegar a representar la instancia de una clase mediante la propiedad *type*. En este caso se puede ver como el recurso Pedro Perez representa una instancia o es de tipo estudiante de “Pre grado”. El dominio y el rango a si mismo pueden ser definidos explícitamente, como se ve en la relación *enseña\_a*, por lo cual se puede decir que para que un recurso enseñe a otro, este debe ser de tipo “*académico*”, mientras que el segundo de tipo “*estudiante*”.

#### 2.2.4.3. OWL

Así como RDFS se creó para dotar de más vocabulario y expresividad semántica sobre RDF. OWL llegó para expandir aún más vocabulario sobre los anteriores, orientado principalmente para mejorar el entendimiento de las descripciones semánticas por las máquinas. OWL es una propuesta de la W3C para un lenguaje formal de ontologías, estandarizado y aceptado, basado en el proyecto DAML+OIL<sup>17</sup> [3]. La ventaja que presenta respecto a otros lenguajes es que permite definir semántica formal que puede ser utilizado por las máquinas para mejorar su capacidad de inferencia. Entre algunos de los aspectos que se pueden expresar usando las primitivas que ofrece OWL están igualdad y desigualdad

<sup>17</sup><http://www.daml.org/2001/03/daml+oil-index.html>

entre clases y propiedades, restricciones más definidas sobre propiedades, características sobre propiedades como similitud, inversa de una propiedad, propiedad simétrica, entre muchas otras [48]. OWL esta divido en 3 sublenguajes que pueden ser utilizados dependiendo de la necesidad de expresividad que se requiera, así como la facilidad computacional para realizar razonamientos. Estos son [40]:

**OWL lite:** Para usuarios que no requieren mucha expresividad. No es totalmente compatible con RDF.

**OWL DL:** Cuando se requiere expresividad sin perder completitud computacional.

**OWL full:** Permite libertad de expresividad por lo que es compatible totalmente con RDF, pero no se asegura la capacidad de procesamiento computacional.

### 2.3. SPARQL

Una vez se encuentran los datos expresados de forma semántica a través de los distintos lenguajes de descripción de ontologías como RDF, RDFS o OWL es evidente la necesidad de acceder y hacer uso de estos datos. Es por esta razón y por la necesidad de un lenguaje estandarizado de consultas sobre RDF que en el año 2007 la W3C define una recomendación para un lenguaje de consultas llamado *SPARQL Protocol and RDF Query Language* (SPARQL) [65]. SPARQL es un lenguaje de consultas para RDF y su acrónimo en español significa Protocolo SPARQL Y Lenguaje de Consulta RDF. Las consultas están compuestas por patrones de tripletas semejantes a los encontrados en RDF, con la diferencia que estos pueden ser variables, los cuales representan el conjunto de valores desconocidos a recuperar. El resultado que se obtiene de las consultas es el subconjunto de datos que se ajusta al patrón dentro del grafo RDF. Al igual que muchos lenguajes de consulta consta de modificadores, que permiten ordenar y filtrar los datos recuperados.

En el segmento de código 2.6 se muestra un ejemplo de consulta simple. La consulta indica que se retorne un elemento dentro del grafo RDF, que cumpla la condición de ser el dominio de la propiedad `enseña_a` en cuyo valor se encuentra “Pedro Perez”. En palabras más simples significa devuélvame el recurso que `enseña_a` Pedro Perez. Por lo que el resultado evidente es Juan Martínez basados en la figura 2.7.

```
1 Select ?academicos
2 WHERE {
3     ?academicos ensena_a Pedro Perez
4 }
```

Segmento de Código 2.6: Consulta de ejemplo en SPARQL

### 2.3.1. SPARQL Endpoint

En el ámbito de los Servicios Web, un endpoint o punto final de conexión define una dirección accesible por un cliente, que sirve como punto conexión para enviar y recibir mensajes de un servicio. En el caso de SPARQL los endpoints son la vía de acceso a repositorios de grafos RDF, lo cual se realiza a través de consultas expresadas en el lenguaje SPARQL. Para aceptar las consultas y retornar los resultados, los endpoints utilizan el protocolo HTTP. Los resultados pueden ser recuperados en varios formatos como XML, *JavaScript Object Notation* (JSON), RDF, HTML.

## 2.4. Estado del arte de la descripción semántica de Servicios Web

Los Servicios Web en la actualidad han llegado a ganar mucha popularidad, lo que ha provocado que se puedan encontrar cientos de ellos en la Web, tanto en formatos SOAP como REST. Esta gran acogida a los servicios que conforme pasa el tiempo sigue creciendo, ha ocasionado que operaciones como búsqueda, autodescripción y composición de servicios se conviertan en operaciones cada vez más solicitadas, ya que amplian en gran medida su funcionalidad y facilitan notablemente su uso. Sin embargo, debido a la naturaleza sintáctica de los servicios que es la forma en que se encuentran descritos la mayoría en la actualidad, se han presentado barreras en la aplicación y creación de dichas operaciones, pues tienen que ser desarrolladas con asistencia de un experto y de forma manual. Como alternativa a dichas carencias dentro de los Servicios Web principalmente en función de impulsar su automatización, muchas propuestas han planteado soluciones que giran en torno a la dotación de semántica a los servicios. La idea de dotar a los Servicios Web con aspectos de la Web Semántica, tienen como objetivo la creación de Servicios Web semánticos, los cuales posean un nivel superior de

información logrando que así sean entendibles tanto por personas como por las máquinas, sean auto descriptivos y por lo tanto permitan la interoperabilidad, composición y descubrimiento de servicios de forma automática [39].

Muchas de las propuestas en el ámbito de la adición de semántica a los Servicios Web han sido enfocadas principalmente sobre SOAP como [62][10][71]. Esto se debe a que SOAP es relativamente más accesible para ser dotado de semántica pues sus datos poseen un formato estructurado como WSDL, además de que fueron los primeros en alcanzar popularidad gracias a la arquitectura orientada a servicios. Sin embargo, como se menciona en la sección 2.1.3 los servicios con Arquitectura en REST han logrado alcanzar también gran popularidad incluso llegando a superar a los servicios SOAP. En este ámbito aunque un poco más rezagado que los servicios SOAP, los servicios REST también han recibido propuestas de varias alternativas para ser dotados de sintaxis y semántica. Entre las alternativas se pueden encontrar desde lenguajes que tratan de especificar formas en las que un servicio REST debe ser descrito para ser entendible semánticamente, hasta alternativas que buscan dotar a la gran cantidad de servicios REST existentes actualmente con los beneficios de la Web Semántica mediante anotaciones especiales.

### 2.4.1. Lenguajes de anotación de Servicios Web

Los lenguajes de anotación de Servicios Web definen la forma en la que se describe un servicio para que puedan ser interpretado fácilmente por las máquinas, ser manipulados y finalmente ser enriquecidos semánticamente. Según el nivel de datos que expresan estos lenguajes se pueden distinguir entre dos tipos: Sintácticos y Semánticos.

En el presente trabajo aunque se tiene un enfoque más orientado hacia descripción semántica de los servicios, se ha visto necesario revisar los dos tipos, pues muchos de los enfoques de lenguajes semánticos que se verán a continuación, toman como base la descripción sintáctica del servicio generada a partir de un lenguaje del mismo tipo.

#### 2.4.1.1. Lenguajes Sintácticos

Los lenguajes sintácticos para los Servicios Web REST proveen de una estructura y expresividad definida que permiten especificar los elementos de un servicio de manera estandarizada. Entre los lenguajes de este tipo se pueden encontrar:

## **hREST**

*hmtl REpresentational State Transfer* (hREST) es un método enfocado en obtener información de las páginas de descripción de los Servicios Web Rest para hacerlos legibles y procesables por computadores. Esta propuesta surge debido a que la mayoría de información acerca de un servicio con arquitectura REST como invocación, entradas, salidas, operaciones se encuentra definidos en una página Web de descripción del servicio. Para obtener la información de dichas páginas, el lenguaje utiliza unas anotaciones especiales sobre las mismas conocidas como microformatos [42]. Una de las características destacables de hREST es que no se requiere separar la información que debe ser procesada por la máquina de la que descripción dedicada al usuario pues las etiquetas (clases) son definidas e integrables con HTML.

## **WADL**

*Web Application Description Language* (WADL) [33] es lenguaje de descripción en XML propuesto por Sun Microsystems. WADL está diseñado para proveer una descripción procesable por las máquinas de aplicaciones Web basadas en HTTP. Este tipo de lenguaje está orientada a la descripción de recursos en lugar de servicios pero es lo suficientemente expresivo para soportarlas. En el ámbito de Servicios Web se puede llegar a interpretar que WADL es para los servicios REST de forma análoga a lo que WSDL es para los servicios SOAP.

## **WSDL 2.0**

WSDL 2.0 [15] extiende las funcionalidades de WSDL 1.1 utilizado para describir servicios SOAP para abarcar también la descripción de servicios HTTP es decir REST. WSDL 2.0 como el WSDL planteado inicialmente 1.1 provee de un modelo y el formato XML para describir formalmente un Servicio Web. Los Servicios Web mediante WSDL son descritos en dos etapas: una abstracta y una concreta. Una de las ventajas de WSDL es la amplia gama de aspectos que se puede definir acerca del servicio que en algunos casos como en la composición de servicios puede ser útil. Sin embargo, algunos autores como Davis John y Rajasree M. S en [39], opinan que esa misma cantidad de información podría perjudicar la simplicidad que ha venido siendo la clave dentro de los servicios REST.

## USDL

*Unified Service Description Language* (USDL) es una propuesta de un lenguaje y modelo de datos para describir los servicios, tanto en sus aspectos técnicos, funcionales y de negocios. A diferencia de otros lenguajes de descripción, se da especial relevancia en aspectos comerciales como precios, condiciones y acuerdos legales, etc. dirigido especialmente a mejorar las funcionalidades requeridas dentro de un mercado de servicios, como por ejemplo descubrimiento, selección e integración de servicios a nuevos entornos [13]. En su versión USDL3M5 (USDL version 3.0 milestone M5) a buscado mejorar su estandarización alineándose con los estándares de la W3C, así como brindando la posibilidad de aplicar los principios de Linked Data y la Web Semántica en su descripción.

### 2.4.1.2. Lenguajes Semánticos

Los lenguajes semánticos para los Servicios Web REST permiten definir los enlaces hacia ontologías y demás recursos semánticos, convirtiendo al Servicio Web en un servicio semánticamente descrito. Entre los más conocidos según la literatura se encuentran:

#### MicroWSMO

*Micro Web Service Modeling Ontology* (MicroWSMO) [41] es una extensión de hREST para añadir anotaciones semánticas a los servicios, para lo cual utiliza una ontología liviana conocida como MicroLite. Esta propuesta recoge cuatro aspectos semánticos que puede tener un servicio para sustentar procesos como descubrimiento, composición, invocación y mediación, como:

**Modelo de información (dominio de la ontología):** Representa los datos especialmente los mensajes de entrada y salida.

**Semántica funcional:** Especifica qué hace el servicio. Considera las precondiciones y sus efectos.

**Comportamiento semántico:** Define la secuencia de operaciones de invocación cuando se invoca un servicio.

**Descripciones no funcionales:** Representa políticas del servicio u otros detalles de la implementación o del ambiente de ejecución del servicio.

## SA-REST

*Semantic Anotation REST* (SA-REST) es una propuesta estandarizada, abierta y flexible para añadir semántica a los Servicios Web de arquitectura REST [69]. SA-REST está basado en hREST y por lo tanto se concentra principalmente en incluir anotaciones semánticas a las páginas de descripción de Servicios Web. Para añadir las anotaciones semánticas a los documentos de descripción de los servicios utiliza un tipo de microformato específico y estandarizado por la W3C conocido como RDFa. RDFa es una técnica que permite la definición de triplets RDF embebidas en documentos XML, HTML, XHTML que permiten la interpretación de la información por parte de las máquinas [35].

## SWSAL

*Semantic Web Services Anotation Language* (SWSAL) es un lenguaje de anotación de Servicios Web, que es utilizado en servicios del tipo SOAP pero que puede ser extendido para realizar la misma función sobre servicios REST [61]. SWSAL para poder ser compatible con REST hace uso de algunas tecnologías adicionales como: WADL utilizado para describir aplicaciones basadas en HTTP. WSML usado para describir una ontología orientada a servicios y SWSAL que es utilizado para enlazar los datos y operaciones definidas en WADL con elementos de WSML.

### 2.4.2. Anotación Semántica de Servicios Web

Todos los lenguajes presentados en la sección anterior, hacen referencia a dotar con semántica al servicio mediante anotaciones especiales conocidas como anotaciones semánticas. Las anotaciones semánticas según los autores de [43] son descritas como el proceso de enlazar recursos electrónicos con ontologías. Este enlace de los recursos de un Servicio Web con los conceptos y elementos de las ontologías en el común de los casos es realizado con el afán de dotar a los Servicios Web con un nuevo nivel de información que permita la automatización de tareas como descubrimiento, invocación y composición mencionados anteriormente. Sin embargo, aunque las anotaciones semánticas buscan la automatización de operaciones relacionadas con los servicios, la creación de servicios semánticos mediante este tipo de anotaciones ha carecido de esta misma característica (automatización).

Gran parte del proceso de anotación semántica que es realizado a los Servicios Web requiere de un importante porcentaje de intervención manual por parte de un experto. Dicho experto es el encargado de entre las múltiples tareas encontrar ontologías para el vocabulario, reconocer los elementos del servicio y desambiguar la información. Sin embargo, cuando se maneja una gran cantidad información como la existente actualmente, dichas tareas llegan a convertirse en una importante barrera de entrada en la adopción de Servicios Web semánticos, ya que aumentan el tiempo que se necesita para describir un servicio y obliga a poseer determinado conocimiento semántico para realizar el proceso de anotación. Para afrontar esta posible barrera de entrada hacia la anotación de servicios se han desarrollado algunos enfoques que buscan reducir la necesidad de intervención humana en el proceso y alcanzar cierto grado de automatización.

#### 2.4.2.1. Anotación Manual

En la anotación manual de Servicios Web el usuario es el responsable de analizar y extraer las características principales del servicio (sintáctica) y enlazarlo con los vocabularios y recursos ontológicos (semántica). En esta forma de anotación el usuario se ve obligado a ejecutar ciertas tareas como el seleccionar el vocabulario adecuado dentro de un gran número de ontologías, desambiguar la información del servicio entre diferentes conceptos que se puedan encontrar y buscar un método de describir formalmente la información. Estas tareas pueden llegar a ser bastante tediosas y propensas a errores.

#### 2.4.2.2. Anotación Asistida

Una alternativa a la búsqueda manual es la de aprovechar la tecnología disponible para crear herramientas que asistan al menos en parte al usuario durante el proceso de anotación. Una de las herramientas que dispone de este enfoque es SWEET.

SWEET [46] es una herramienta para la creación de servicios RESTful Semánticos, que soporta la creación de descripciones legibles por las máquinas mediante la adición de anotaciones semánticas en busca de presentar mejoras en el descubrimiento de servicios, creación de mashups y la invocación. Para la creación de servicios semánticos SWEET se basa en tecnologías como hREST (sección 2.4.1.1) que es utilizado en el proceso de obtención de microformatos sobre la descripción de servicios y MicroWSMO (sección 2.4.1.2) para la anotación semántica de propiedades.

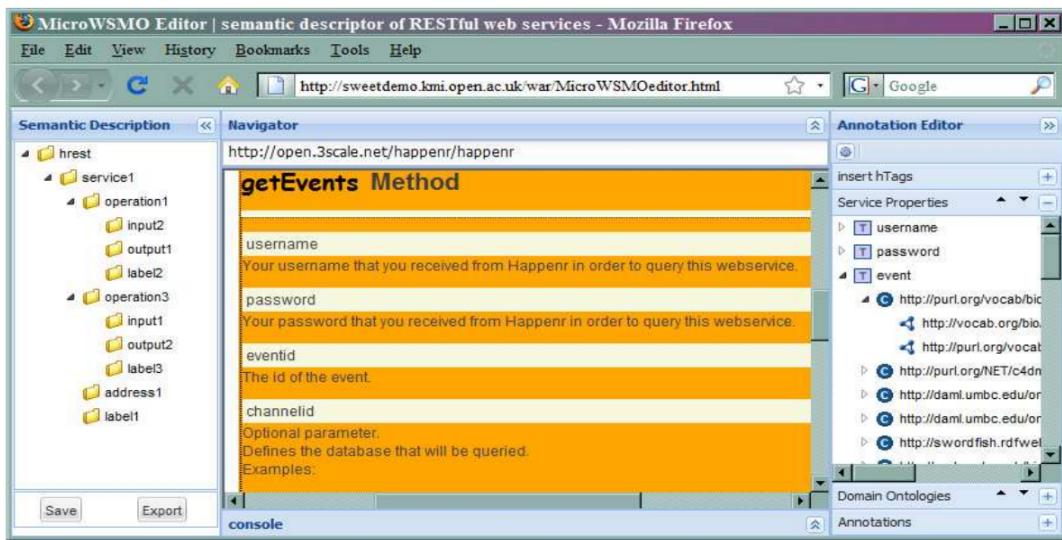


Figura 2.8: Interfaz SWEET [45]

SWEET dispone de una aplicación Web que con la ayuda de JavaScript<sup>18</sup> y ExtGWT<sup>19</sup>, es la encargada de tomar la página de descripción del servicio y presentarla al usuario para facilitar el proceso de anotación (figura 2.8 ). El proceso de anotación dentro de SWEET consiste básicamente en ofrecer al usuario la posibilidad de reconocer dentro del documento de descripción del servicio los elementos relevantes que pertenecen al servicio (operaciones, entradas, salidas) para posteriormente vincularlos con vocabularios semánticos. Para lograr la vinculación con un vocabulario previamente seleccionado, SWEET hace uso del buscador semántico WATSON embebido dentro de la aplicación. Al finalizar el proceso de anotación SWEET puede convertir el documento HTML a RDF MicroWSMO, el cual puede ser manipulado y utilizado como recurso semántico y descriptivo del Servicio Web. La ventaja de dicha propuesta es que no se requiere tener conocimientos avanzados acerca de anotación semántica de servicios.

#### 2.4.2.3. Anotación Automática

Una propuesta dirigida hacia la anotación automática de servicios se puede encontrar en la tesis de Saquicela[21]. En esta propuesta el autor plantea algunas etapas y procedimientos para lograr la automatización parcial del proceso de anotación semántica de servicios apoyándose para eso en la inclusión de recursos

<sup>18</sup>JavaScript: Es un lenguaje script multi-paradigma, basado en prototipos, dinámico que soporta estilos de programación funcional, orientada a objetos e imperativa.

<sup>19</sup><http://gwt-ext.com/>

externos. Las etapas planteadas por dicha propuesta se pueden ver en la figura 2.9, en el cual se describe el proceso de anotación del API de un Servicio Web de tipo geográfico hasta alcanzar su anotación semántica.

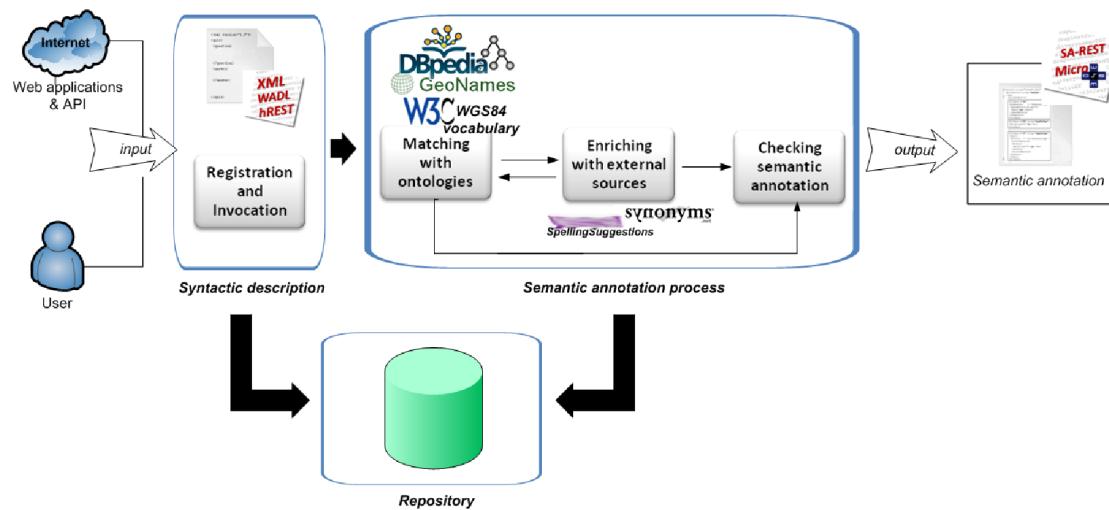


Figura 2.9: Etapas del sistema de anotación automática [21]

El procedimiento que sigue esta aproximación para la anotación semántica comienza con la etapa de recopilación de la descripción sintáctica del servicio. Para la obtención de la descripción sintáctica del servicio se utiliza en el caso de servicios basados en SOAP el archivo WSDL en el que consta la información necesaria para la invocación del servicio mientras que para servicios basados en REST, son los datos provenientes de la URL y del XML de respuesta. Los datos que son recuperados del servicio que contienen tanto las entradas, salidas, URL de invocación y metadatos son almacenados en un modelo de base de datos planteado por el autor. Una vez registrados los elementos del servicio se prosiguen con una etapa de enriquecimiento de parámetros. El proceso de enriquecimiento hace referencia al uso de recursos externos y se utiliza con el fin de aumentar el número de elementos a comparar con las ontologías para así alcanzar una posibilidad mayor de anotación durante la etapa de *matching*. La etapa de *matching* en la propuesta, consiste en una etapa de emparejamiento en donde se mide la similaridad entre los elementos de la ontología con los elementos de la descripción del servicio enriquecido para encontrar así sus elementos en común. Finalmente, al entrar en la etapa de validación de anotaciones semánticas se comprueba si las instancias de las anotaciones realizadas sobre los parámetros del servicio tanto entradas y salidas enriquecidas son invocables, confirmando que la anotación pertenece al mismo tipo que la indicada en el servicio.

En el proceso de anotación semántica de servicios descrito anteriormente se puede notar que no existe una intervención directa por parte de los usuarios dentro del proceso de anotación y únicamente es necesario de su participación en el proceso de verificación de resultados. Dicha intervención reducida por parte de los usuarios que se encamina hacia la automatización del proceso de anotación, concuerda con los objetivos de la presente tesis por lo que la misma ha sido utilizado como base de esta propuesta. Sin embargo, como toda aproximación que se encuentra en investigación, posee algunas falencias que limitan su total automatización, como la necesidad de conocer a priori el dominio del servicio para incluir dentro del proceso la ontología adecuada. Otro de las debilidades de la propuesta planteada es la carencia de una herramienta que implemente el método, pues sin ella el proceso de anotación planteado no puede probarse exhaustivamente dentro de otros dominios.

## 2.5. Tecnologías para la Búsqueda y Selección de Ontologías

Cuando se presenta la necesidad de usar ontologías como por ejemplo cuando se está describiendo documentos semánticos (RDF) o en el caso de requerir añadir anotaciones semánticas, muchas veces no se tiene el conocimiento específico de la ontología que se necesita usar para cubrir el vocabulario que se está manejando. Este problema puede presentarse en el enfoque de anotación automático descrito en la sección 2.4.2.3 pues aún requiere de la selección manual de ontologías para el servicio que quiere anotarse. También es uno de los principales problemas a resolverse en la selección automática de ontologías que se ha planteado implementarse (capítulo 4). Por lo que para enfrentar a este problema es necesario buscar alternativas que permitan obtener una ontología propicia para dotar de semántica a lo que se esté tratando. Por lo general para poder obtener una ontología se puede seguir una de las siguientes opciones : 1) Realizar una búsqueda de ontologías en un repositorio ontológico en base a la información existente; 2) Plantearse la creación de una ontología propia. En muchos casos se puede considerar que es necesario aplicar la segunda opción, es decir la creación de ontologías desde cero, sin embargo es recomendable realizar previamente una búsqueda de ontologías existentes previo a su generación. La necesidad de buscar una ontología para ser utilizada en lugar de la creación de una nueva, obedece a la naturaleza de la Web Semántica que impulsan al reusó de información y al enlace con recursos semánti-

cos existentes para ampliar el conocimiento [19]. Aunque desde esa perspectiva la búsqueda de ontologías se convierte en un paso obligatorio, esta no ha dejado de ser un proceso complejo. Dicha complejidad está dada por el crecimiento de la Web Semántica, donde se añaden cada vez más ontologías y se incorporan nuevos dominios. Para afrontar el problema de la búsqueda en este gran conjunto de datos ontológicos se han planteado algunas soluciones, entre las que se encuentran la búsqueda con ayuda de motores semánticos, populares debido a su facilidad de uso y por poder ser manejados por personas no expertas y otras técnicas más avanzadas que al requerir características específicas no soportadas por los buscadores, pueden necesitar ser desarrolladas.

### 2.5.1. Motores de Búsqueda Semánticos de Ontologías

Una de las opciones que habitualmente se utiliza cuando se requiere realizar búsquedas en la Web, son los motores de búsqueda y en la Web Semántica esta opción también está empezando a ser bastante reconocida. Los motores de búsqueda a los que se está acostumbrado en la Web convencional, permiten realizar búsquedas basadas en palabras clave, lo que permite obtener resultados inmediatos aún cuando se carezcan de datos exactos de lo que se pretende encontrar. En el ámbito de la Web Semántica se ha mantenido esta misma idea y con el fin de facilitar las búsquedas de recursos sobre un gran conjunto de datos semánticos, se crearon los motores de búsqueda semánticos. Según la definición presentada por la W3C un motor de búsqueda semántico es una aplicación para encontrar ontologías que requieren un esfuerzo razonable, ya que las consultas son usualmente escritas en palabras de lenguaje natural y sus resultados son presentados en base a una puntuación. Además, tienen la capacidad de proveer información adicional sobre los resultados o metadatos [83]. Los motores de búsqueda semánticos por lo tanto se puede apreciar que son semejantes a los motores de búsquedas de la Web normal pero que tienen como fin encontrar conceptos semánticos y ontologías.

Existen varios buscadores semánticos actualmente, algunos de los cuales son orientados a buscar conceptos, mientras que otros proveen ontologías y unos cuantos proveen ambas funciones. Entre los motores de búsqueda más conocidos se encuentran:

- FalconS
- Watson

En la sección 4.3 se describen cada uno de los buscadores semánticos y sus características.

### 2.5.2. Técnicas de búsqueda de ontologías basadas en análisis de texto

Frente a la necesidad de encontrar una ontología con características especiales que no cubren otras opciones como los buscadores semánticos, se puede recurrir a una implementación de un método de búsqueda propio. Estos métodos de búsqueda al igual que los buscadores semánticos por lo general deben tener la capacidad de encontrar una ontología tomando como base a un conjunto de palabras, en la práctica esto significa, que de entre un conjunto determinado de ontologías, los métodos de búsqueda deben proveer la ontología que tenga el mayor número de coincidencias o similitud con las palabras ingresadas. Para determinar la similitud entre las palabras ingresadas y los elementos de la ontología (*matching*) se puede tomar varios enfoques como los tratados en [72] que son : emparejamiento lógico, emparejamiento basado en sintaxis, emparejamiento de estructura semántica. Sin embargo, considerando que la información semántica es precisamente de lo que carecen las palabras ingresadas, la aproximación de emparejamiento basado en sintaxis llega a ser la más adecuada. Una ventaja de tomar el enfoque sintáctico consiste en que tanto las palabras ingresadas para la búsqueda como la ontología son consideradas como conjunto de palabras, simplificando el problema y convirtiéndose de esta manera en un problema de similitud entre textos. Los problemas de similitud entre textos son bastantes conocidos por ejemplo en el ámbito de clasificación de texto, autocompletado, reconocimiento de idioma, etc. Por lo que los métodos de similitud entre textos ofrecen muchas alternativas y aproximaciones de solución. Entre dichas aproximaciones se encuentran soluciones que van desde la simple comparación de igualdad y similitud entre elementos (*matching*), *clustering*, hasta uso de herramientas avanzadas de búsqueda de texto e indexado como Solr<sup>20</sup> o MySQL<sup>21</sup> Fulltext. A continuación se expondrá una introducción de cada una de las técnicas con las que se pretende solucionar el problema de búsquedas de ontologías, utilizadas en el desarrollo de este trabajo.

---

<sup>20</sup><http://lucene.apache.org/solr/>

<sup>21</sup><http://www.mysql.com/>

### 2.5.2.1. Emparejamiento entre textos

El emparejamiento entre textos consiste en comparar cada uno de los elementos o palabras de un texto con el otro. Para la comparación se pueden utilizar medidas de igualdad o de similitud. Las medidas de igualdad determinan si una palabra es exactamente igual a otra, lo que se puede comprobar comparando cada una de sus letras. Los resultados de las medidas de la igualdad son discretos es decir 1 o 0, dependiendo de son iguales o no. En el caso de las medidas de similitud se pretende determinar cuan cercana se encuentra una palabra de la otra y utilizan valores que pueden ir desde 0 a 1 de acuerdo al nivel de similitud que presenten. Las medidas de similitud se pueden implementar usando diferentes algoritmos, algunos de los cuales se encuentran detallados en la tabla 2.2.

Medidas similitud	Descripción
Jaro Similarity	Similitud de cadenas basada en la métrica de distancia Jaro. Toma en cuenta pequeñas desviaciones que ocurren al escribir.
Jaro Winkler	Medida de similitud entre dos cadenas, variante de la métrica Jaro. Esta extensión modifica los pesos de los pares que se emparejan mal y que comparten un prefijo en común.
Levenshtein Distance	Mínimo número de edición necesario para transformar una cadena en otra.
Monge Elkan Distance	Medida de similitud basada en la métrica Monge Elkan
Needleman-Wunch Distance	Realiza alineamiento global entre dos cadenas
Smith – Waterman - Gotoh	Realiza alineamiento local entre dos cadenas.

Tabla 2.2: Métricas de similitud [9]

### 2.5.2.2. *Clustering* de texto

La clusterización de texto es una técnica que permite agrupar y organizar una cantidad grande de texto en un número pequeño de clústers significativos y coherentes [53], que facilitan la navegación y mecanismos de búsqueda. La clusterización de texto puede tener varios niveles de granularidad por ejemplo realizarse la clusterización a nivel documentos, oraciones, términos, entre otros [1]. Esta técnica es utilizada principalmente en tareas relacionadas con minería de datos para encontrar información implícita, pero su uso se ha ampliado a organización y búsquedas de documentos, resumen de texto, extracción de tópicos, clasificación de documentos, entre otras aplicaciones.

### 2.5.2.2.1. Preprocesamiento de Texto

Para el desarrollo de algunas técnicas de procesamiento de texto como el de clusterización, se requiere que los datos con los que se está tratando pasen previamente por un preprocesamiento. El preprocesamiento es necesario para asegurar la obtención óptima de resultados y evitar factores que no aporten al proceso o puedan presentar ruido. Entre las consideraciones más comunes en esta etapa se encuentran:

**Conversión a tokens:** Los documentos deben ser representados como unidades que puedan ser procesadas por los algoritmos. Por lo general se suele separar por palabras, pero dependiendo de la granularidad deseada se puede dividir por frases u oraciones, entre otras.

**Eliminación de stopwords:** Las stopwords son palabras que no son representativas y que aparecen continuamente en los textos formadas comúnmente por pronombres, preposiciones como “el”, “los”, “a”, “en”, entre otros. En algunos casos, para eliminar el ruido y datos innecesarios de los datos se suele excluir dichas palabras.

**Lematización(*Stemming*):** Consiste en la normalización de las palabras mediante un análisis morfológico con el fin de reducir las palabras a formas más básica o raíz. Suele utilizarse en casos donde una palabra necesita identificarse como tal, independientemente de su conjugación, diminutivo u otras variaciones.

Otro de los pasos que son necesarios previo a la aplicación de la técnica de clusterización es la conversión del documento a un vector de términos que puedan ser fácilmente manejados por el cluster, conocido como vocabulario. Dependiendo de la técnica de representación de documentos los pesos de las palabras puede variar por ejemplo:

**Frecuencia de términos (TF):** Considera más relevantes las palabras que más se repiten

**Frecuencia inversa de términos(IDF):** Considera que mientras mayor número de veces aparece la palabra menor será su peso.

**TF-IDF:** Considera ambas situaciones para asignar el peso.

### 2.5.2.2.2. Métricas de los clústeres

Los clusters para agrupar las elementos de texto (documentos, oraciones, términos) utilizan por lo general medidas de similitud o distancia. Las medidas de similitud y distancia permiten establecer cuán cercano se encuentran los elementos del centro del cluster y por lo tanto cuán cercanos se encuentran los elementos entre sí. Conocer la distancia entre sus elementos y el centro del cluster permite determinar si el elemento pertenece a un cluster o debe asignarse a otro. Las medidas más conocidas se detallan en la tabla 2.3.

Algoritmo de similitud o distancia	Descripción
Distancia Euclídea	Es una de las más populares medidas y tiene como objetivo medir la distancia entre dos puntos de forma geométrica.
Similitud del Coseno	Usada comúnmente en problemas de texto y se caracteriza por calcular la similitud independiente del tamaño del texto.
Coeficiente de Jaccard	Compara la suma de los términos compartidos con la suma de los términos no compartidos en los documentos.
Correlación de Pearson	Encuentra el número óptimo de clústeres, basado en la similitud entre los datos.

Tabla 2.3: Métricas de similitud y distancia comunes en el ámbito de clusterización

### 2.5.2.2.3. Técnicas de clústeres

Conjuntamente con las funciones de similitud o distancia son utilizadas varias técnicas de *clustering*, que definen el tipo de cluster que se está realizando. Las técnicas de *clustering* determinan el proceso por el que pasaran los clústeres antes de llegar a presentar el resultado final. Entre estas técnicas de *clustering* se pueden encontrar dos principales:

#### Técnica de clusterización por jerarquías

Esta técnica produce particiones anidadas de clusters, que se presentan con una estructura de árbol conocido como dendograma. Los nodos hojas son considerados clústeres individuales mientras que los que se acerca a la raíz están formados por la conjunción de clústeres, de nivel inferior. Se puede apreciar la representación de un cluster jerárquico en la figura 2.10.

Existen dos aproximaciones que permiten generar clústeres jerarquizados:

**Aglomerativo:** Comienzan con clústeres individuales en cada paso se unen los clústeres más cercanos produciendo uno nuevo.

**Divisivo:** Comienza con un cluster grande que engloba todos los clusters y en cada paso este se va diviendo en un par de clústeres.

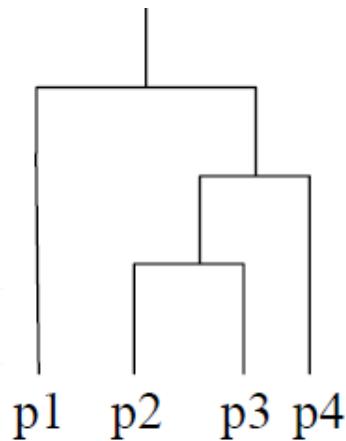


Figura 2.10: Representación de la clusterización por jerarquía [74]

## Técnicas de clustering por particiones

Esta técnica consiste en determinar inicialmente  $k$  clústeres o particiones de una sola vez, en contraste con los clústeres jerárquicos. Con los pasos posteriores estos se van afinando para que los datos dentro del cluster sean mejor representados. El más conocido de los clústeres de este tipo es el k-means. En la figura 2.11 se puede ver una representación de dos clústeres y sus centroides representados con una cruz.

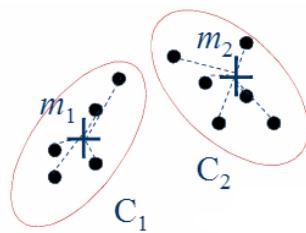


Figura 2.11: Representación de la clusterización por agrupación [8]

Los clusters basados en k-means consiste en un algoritmo que toma en consideración un  $k$  número de puntos iniciales del cual toma su nombre. Este algoritmo pretende encontrar para  $n$  puntos de un espacio dimensional, los  $K$  centros dentro de ese mismo espacio dimensional para los cuales la distancia para cada uno de los puntos hacia los centros más cercanos  $K$  se minimice. Para medir la distancia entre los puntos y los  $k$  centros se necesitan una función o métrica, en la que

se puede utilizar algunas de las técnicas presentadas en la tabla 2.3, pero por lo general suele utilizarse la distancia euclídea para este tipo de cluster.

La distancia euclídea es la distancia geométrica entre dos puntos en el espacio dada por fórmula:

$$Dist(di, dj) = ||di - dj||$$

En el caso de los documentos o procesamiento de texto, dados dos documentos  $ta$  y  $tb$  representados por sus vectores de términos. La distancia euclídea entre los dos documentos está definida como [38]:

$$De(\vec{ta}, \vec{tb}) = (\sum_{t=1}^m |W_{t,a} - W_{t,b}|^2)^{1/2}$$

Donde el conjunto de términos es  $T = t1, \dots, tm$ , mientras que  $W$  representa los valores de los pesos obtenidos mediante TFIDF. Dicho de otra forma, la formula representa la suma de cada una de las distancias euclídeas obtenidas entre los pesos de cada uno de los términos pertenecientes a los documentos comparados.

### 2.5.3. MySQL FullText

MySQL Fulltext es una característica especial que provee el motor de base de datos MySQL para indexado y búsqueda de texto [55]. Esta característica fue creada al encontrarse con algunas necesidades sobre los textos que no son satisfechas por los métodos de búsqueda convencionales por medio de consultas SQL. Entre estas se encuentran.

- Rendimiento afectado al utilizar sentencias *Like* que realiza una búsqueda sobre toda una tabla.
- Carencias de búsquedas flexibles por ejemplo que permitan buscar un resultado que contenga una palabra pero no otra.
- Dificultad en determinar el orden de relevancia que tienen las búsquedas.

Para enfrentarse a esas necesidades MySQL provee la función MySQL Fulltext, que pretende brindar las ventajas de los métodos de búsqueda similares a los que nos ofrece un buscador Web como búsquedas en lenguaje natural y respuestas inmediatas.

### 2.5.3.1. Arquitectura general

La arquitectura de herramientas que ayudan a la indexación y búsqueda de texto como MySQL Fulltext se puede observar en la figura 2.12. Inicialmente el proceso de indexación que emplea MySQL Fulltext según su arquitectura, requiere de un conjunto de pasos de preprocesamiento similar a lo descrito en el proceso de clusterización (sección 2.5.2.2), en los que consta : dividir el documento en pequeñas unidades conocidas como tokens (*tokenizer*), y remover palabras que no aportan información relevante (*stopword*). Adicionalmente, a los procesos mencionados se debe cumplir el proceso de indexación. La indexación es la característica que brinda un acceso rápido a los datos y dentro de MySQL Fulltext contiene un diseño de índices invertidos. Los índices invertidos consisten en almacenar una lista de palabras en una tabla de índices y por cada palabra almacenar una lista de documentos en las que aparece. Para apoyar la búsqueda de proximidad, la información de posición para cada palabra también se almacena, como desplazamiento de bytes [56].

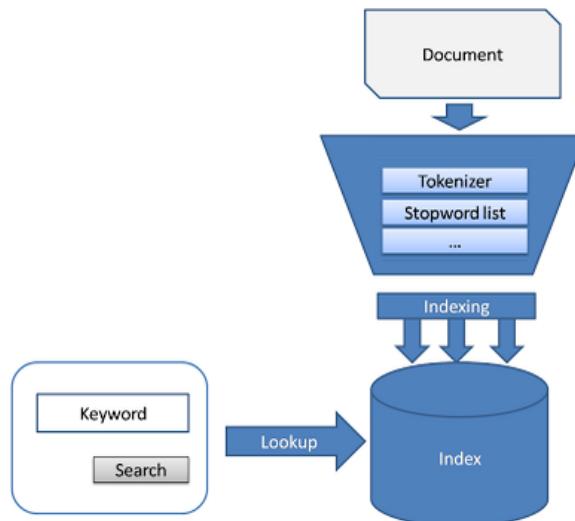


Figura 2.12: Arquitectura general de MySQL Fulltext [59]

### 2.5.3.2. Ranking en MySQL Fulltext

MySQL Fulltext provee entre sus múltiples ventajas la organización de resultados por relevancia. La organización de los resultados por relevancia permite determinar qué tan bueno es la coincidencia de la búsqueda con respecto a los datos que se desean recuperar. Dicha característica es bastante útil en muchas

situaciones para determinar cuáles son los mejores resultados encontrados. Para poder realizar dicha organización MySQL Fulltext utiliza un sistema conocido como Vector Espacial. El Vector Espacial es una representación de un sistema de líneas que se extienden en diferentes dimensiones (una dimensión por término) que varían en la distancia (una unidad de distancia por aparición de término). Esta representación es ventajosa, ya que si las ocurrencias de los términos son líneas en espacio multidimensionales, se puede aplicar trigonometría básica para calcular las distancias, que es equivalentes a calcular medidas de similitud. En la práctica a la distancias se les refiere como pesos, y en MySQL Fulltext calcula los pesos de la siguiente manera [54].

$$w = (\log(dt_f) + 1) / \sum dt_f * U / (1 + 0,0115 * U) * \log((N - nf) / nf)$$

Donde w es el peso y los demás elementos representan:

*dt<sub>f</sub>* Es el número de veces que aparece el término en el documento

*Sumdt<sub>f</sub>* Suma de  $\log(dt_f) + 1$  para todos los términos en el mismo documento.

*U* Número de términos únicos en el documento

*N* Número total de términos

*Nf* Número de documentos que contiene el término

La primera parte  $(\log(dt_f) + 1) / \sum dt_f$  es la parte base. La segunda parte  $U / (1 + 0,0115 * U)$  es un factor de normalización. La tercera parte  $\log((N - nf) / nf)$  comprende un multiplicador global, y trata de hacer una mejor estimación de la probabilidad de que un término será relevante.

La parte base por el factor de normalización es lo que comúnmente se almacena en la tabla de índices de MySQL.

Para calcular el ranking se utiliza la siguiente formula

$$R = w * qf$$

**W** es el peso

**Qf** es el número de veces que el término aparece en la consulta.

### 2.5.3.3. Búsquedas con MySQL Fulltext

Con MySQL Fulltext están disponibles tres tipos de búsqueda:

**Búsqueda booleana:** Utiliza reglas especiales de consulta junto con el texto de búsqueda. Por ejemplo se puede utilizar operadores para indicar si una palabra debe aparecer, mientras que otra debe estar ausente o búsquedas donde una palabra tiene más relevancia que otra.

**Búsqueda por lenguaje natural:** El texto ingresado es considerado como una frase en lenguaje natural. No se consideran operadores especiales. Se distingue este método porque analiza las palabras en todas las columnas y si se encuentran presentes más de un 50 % de las columnas analizadas estas no son consideradas.

**Expansión de consulta:** Es una extensión de la búsqueda por lenguaje natural y se caracteriza por que usa las filas más relevantes devueltas por la primera búsqueda, para agregarse a la cadena de búsqueda por segunda vez. La consulta devuelve las filas de resultado obtenidas de la segunda consulta.

### 2.5.4. Apache Solr

Solr es un motor de búsqueda empresarial de código abierto, que contiene embebido al proyecto Lucene como parte central de la solución para la indexación y recuperación de documentos. Tanto Solr como Lucene están basados en JAVA y juntos proporcionan un motor de búsqueda altamente fiable, escalable y tolerante a fallos, que provee además otras características apreciables como indexación distribuida, replicación de datos, configuración centralizada, entre otras. Para los usuarios Solr provee algunas funcionalidades similares a los de los motores de búsqueda en la Web como [28]:

**Paginación y ordenamiento:** Evita la sobrecarga de resultados devolviendo un número determinado de resultados por páginas ordenando relevancia.

**Facetas (*Faceting*):** Proporciona a los usuarios herramientas para refinar sus criterios de búsqueda y descubrir más información mediante la categorización de los resultados de búsqueda en subgrupos utilizando facetas.

**Auto-sugerencias:** Permite completar lo que están escribiendo los usuarios al presentar una lista de términos y frases sugeridas sobre la base de documentos en su índice.

**Corrección de Errores :** Permite hacer correcciones sobre términos mal escritos y dar sugerencias con palabras que proveen mejores resultados.

**Búsquedas Geográficas:** Permite manejar longitud y latitud para realizar búsquedas en torno a distancias geográficas.

#### 2.5.4.1. Arquitectura General Solr

En la figura 2.13 se puede apreciar la arquitectura general de Solr. Solr funciona como una aplicación de Java para la Web, que puede correr sobre un motor de Servlets de Java (Jetty Tomcat). Como núcleo para la indexación de documentos y la consulta de documentos utiliza Lucene, al cual se le añaden algunas mejoras para su utilización. Esta mejora consiste en que Solr permite configurar los archivos para la indexación por medio de archivos en XML, que evita la necesidad de desarrollar directamente sobre código Java, como sería necesario si únicamente se utilizará Lucene.

Solr contiene algunos componentes dentro de su arquitectura que son los encargados de brindar toda la funcionalidad descrita en la sección anterior, estos componentes son:

**Componente de Procesamiento de Consultas y Captura:** Procesa las consultas y añade las características descritas como autocompletado, autocorrección, etc.

**Componente de Procesamiento de Actualizaciones:** Este componente administra la adición de nuevos documentos, su modificación o borrado.

**Componente de Análisis de Texto:** Componente que permite el proceso de análisis de texto previo a la indexación para la adecuación de los datos. En esta etapa de forma similar a los métodos anteriores se realiza tareas de transformación de datos como stemming, división en tokens, adicción de sinónimos, etc.

**Índices Distribuidos de Lucene:** Lucene permite el uso de múltiples índices distribuidos para aumentar la velocidad de búsqueda con lo cual se puede utilizar múltiples hilos de búsqueda. También maneja replicas en el caso de que se tenga una carga elevada de consultas.

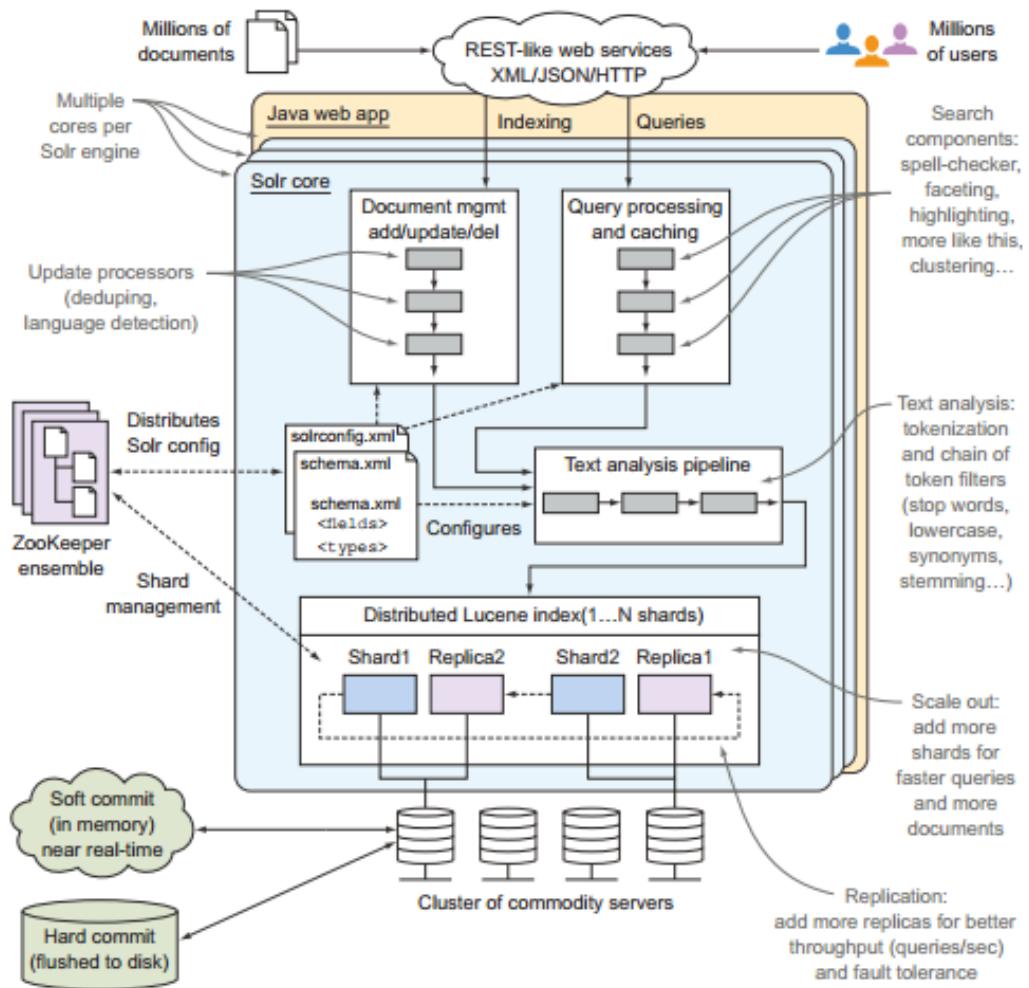


Figura 2.13: Arquitectura general de Solr [28]

#### 2.5.4.2. Indexado y representación de los datos

El indexado dentro de Solr está dado por Lucene, que proporciona la capacidad de construcción y gestión de índices invertidos similar a los utilizados por MySQL Fulltext. Como unidad básica para la búsqueda e indexado se utiliza una estructura especial conocida como documento que está formado por una colección de campos (no anidados).

Para definir cuáles son los campos que contendrá el documento, primeramente se debe crear un archivo XML con el esquema. El esquema describe algunas características respecto a los documentos que se ingresaran como [78]:

- Los tipo de campos
- Los campos que sirven como llaves.
- Los campos que son requeridos
- Los campos que deben indexarse y almacenarse
- Definición de nuevos campos con ayuda de filtros

En el esquema los tokenizadores y filtros que se aplicarán se definen por campos, tanto para la búsqueda como para la indexación.

#### 2.5.4.3. Ranking en Solr

Lucene para obtener los puntajes dentro de los índices combina el modelo de Espacio Vectorial (VSN) con el modelo booleano (BM). La fórmula que aplica es la siguiente [4]:

$$score(q, d) = coord(q, d)*queryNorm(q)* \sum_{tfenq} (tf(tend)*idf(t)^2*t.getBoost()*norm(d, t))$$

Donde *score* hace referencia al puntaje de la consulta en los documentos y los demás elementos:

**Tf (t in d):** Número de veces que el término t aparece en el documento d.

**Idf (t) :** Frecuencia inversa del documento. El número de documentos en los que aparece el término.

**Cood(q,d):** Factor de puntaje que calcula el número de veces que los términos de la consulta aparecen en el documento.

**queryNorm (q):** Es un factor de normalización para que el puntaje de las consultas pueda ser comparable.

**t.getBoost ()** Considera al boost<sup>22</sup> que contiene el término t en la consulta q.

**norm(t,d)** Normalización que considera la extensión del documento y los boost declarados al momento de hacer la indexación.

#### 2.5.4.4. Búsqueda en Solr

Las búsquedas en Solr son realizados mediante lenguaje natural, sin embargo también soporta sintaxis propia de Lucene que permite realizar búsquedas avanzadas. Entre los caracteres que soporta están presentados en la tabla 2.4:

Carácter	Descripción	Ejemplo
:	Sirve para especificar el campo sobre el que se realizará la búsqueda y el valor	Campo1:dato1
AND, OR	Permite utilizar operadores lógicos	Campo1:dato1 or Campo2:dato2
*	Indica que la búsqueda puede comenzar o terminar con el valor ingresado.	fu*ol (Comienza con fu y termina en ol, podría retornar futbol)
~	Coincidencia por proximidad, permite indicar a cuantas palabras se encuentra la primera palabra de la segunda.	(dato1 ~ dato2) n

Tabla 2.4: Ejemplo de caracteres especiales usados en las búsquedas de Solr

## 2.6. Herramientas de integración

Una de las principales características que se busca cuando se desarrolla una solución de software que tiene como objetivo ser ampliamente usada en múltiples entornos, es que sea compatible e integrable con diversos sistemas. Esto se debe a que al permitirse la integración de una solución con sistemas más complejos, ésta puede llegar a ser más fácilmente adoptada ya sea utilizando directamente su funcionalidad con herramientas existentes o desarrollando una nueva a partir de estas. Por esta razón dentro del desarrollo del prototipo planteado se ha visto pertinente revisar algunas de las herramientas de integración orientadas a servicios existentes que puedan contribuir a mejorar la escalabilidad de la solución de software, con miras a su expansión dentro de proyectos futuros.

<sup>22</sup>Boost es un potenciador de puntaje de los términos o documentos, que puede ser declarado explícitamente para aumentar relevancia de un elemento en los índices o en la búsqueda.

Dado que la mayoría de herramientas de integración orientadas a servicios se originaron frente a necesidades en el ámbito empresarial, se abordará el tema desde esta perspectiva, tratando primeramente las tecnologías que fueron usados como mecanismos de integración y que dieron paso a una de las más conocidas en la actualidad como son los Buses de Servicios Empresariales. Esta tecnología es de interés pues como se describe en la sección 1.3, ha sido planteada como marco de desarrollo para probar los conceptos aplicados al presente proyecto de tesis.

### **2.6.1. Integración de Software en Ambientes Empresariales**

Las empresas continuamente se encuentran en crecimiento y para afrontar los rápidos cambios en el mercado, así como la aparición de nuevos requerimientos, adquieren diferentes productos de software conforme las necesidades requieren ser atendidas. Por ejemplo se puede llegar a requerir débito automático para tarjetas de crédito, notificaciones electrónicas, entre tantas otras. No obstante, aunque muchos de estos productos pueden llegar a cumplir su función de manera efectiva, pueden no ser capaces de llegar a trabajar de manera totalmente integrada con el resto del sistema. En la mayoría de los casos esto se debe a que cada uno de dichos productos de software o aplicaciones son desarrollados por determinados proveedores, los que a su vez utilizan tecnología específica para este fin. Adicionalmente dado que cada empresa posee su propio entorno de TI, se requieren de consideraciones especiales para asegurar su compatibilidad e integración con el resto de componentes, ya que de otro modo daría como resultado un conjunto de elementos funcionales pero de manera independiente. Sin embargo, frente a la tendencia de las empresas que requieren la rápida adaptación a los cambios en este mundo competitivo, el aislamiento y heterogeneidad de aplicaciones se han convertido en un obstáculo a superarse en el aprovechamiento óptimo de los datos para la ágil toma de decisiones. Entre las soluciones que se pueden encontrar la superar la heterogeneidad de aplicaciones y datos, como primer acercamiento se encuentra la creación de una nueva herramienta que englobe a todas las aplicaciones y por otro lado el usar tecnologías de integración. En el primer caso la capacidad de desarrollar una herramienta aunque es una solución que podría terminar efectivamente con el problema de integración, puede llegar a ser exageradamente costosa debido al número de requerimientos que debe cubrir. Además dichas funcionalidades están estrechamente ligadas, lo que cambiarlas para adap-

tarse a nuevos requerimientos puede llegar a tener complicaciones adicionales. En el caso de tecnologías de integración se tiene como objetivo conseguir la interoperabilidad de diferentes aplicaciones, independientemente de la tecnología con la que fueron desarrollados. De esta manera se puede aprovechar las funcionalidades de cada uno de sus elementos o incluso llegar a generar nuevas funcionalidades, de modo que los usuarios puedan llegar a percibir el conjunto de aplicaciones integradas como si se tratase de una sola (generación de vistas) [20].

Existen un conjunto de tecnologías que han llegado a causar especial interés en el área de integración como son la *Arquitectura Orientada a Servicios* (SOA), *Integración de Aplicaciones Empresariales* (EIA), *Business-to-Business* (B2B), y Servicios Web. Tecnologías principalmente enfocadas en mejorar los resultados y aumentar el valor de los procesos de negocio integrados [14]. Otra de las tecnologías de integración que más ha llamado la atención ultimamente es el *Bus de Servicios Empresarial* (ESB), que reúne algunos de los mejores aspectos de tecnologías EAI como SOA, los cuales se expondrán en el siguiente punto luego de mostrar las características que poseen estos dos exponentes de la integración.

### 2.6.1.1. Arquitectura Orientada a Servicios

Una arquitectura orientada a servicios es un enfoque en donde los recursos de software son presentados como unidades visibles, reutilizables y disponibles dentro de la Web conocidos como servicios [25]. Estos servicios son autocontenido y se puede acceder a ellos por medio de interfaces estandarizadas. Esta propuesta nació inicialmente ante la necesidad de una arquitectura de sistemas distribuidos que dio paso a tecnologías como CORBA, RMI, etc[5]. Con la evolución de la Web los SOA respondieron usando para esto tecnologías de Servicios Web como REST y SOAP, mejorando la estandarización de interfaces por protocolos y formatos abiertos. SOA aparece como una vía de integración, debido a que en ambos casos tanto en los sistemas distribuidos como los sistemas de integración requieren el trabajo conjunto de diferentes aplicaciones, capaces de funcionar de forma independiente de la plataforma en la que se encuentra, del lenguaje de programación y con bajo acoplamiento entre sí.

### 2.6.1.2. Integración de Aplicaciones Empresariales (EIA)

Conforme la necesidad de mecanismos de integración se hicieron más evidentes, se desarrollaron un conjunto de herramientas, así como una arquitectura específica para este fin conocido como EIA. EIA tiene como objetivo que las diver-

sas aplicaciones en una empresa así como sistemas asociados se comuniquen entre si, independientemente de la plataforma, lenguaje y la localización geográfica en la que se encuentran. Para lograr dichos objetivos EAI utiliza un componente llamado *Message Broker* como parte central del entorno de integración. *Message Broker* es un sistema de cola de mensajes que permite almacenar, procesar y enviar mensajes de forma asíncrona.

Una de las capacidades destacadas por parte de EAI es que por medio de la manipulación de los mensajes se puede llegar ha realizarse transformación, traducción, ruteo sobre los mensajes, según el proceso definido por la empresa o según las necesidades de la aplicación destinataria [25]. Existen dos tipos de arquitecturas con EAI que son la HUB/SPOKE y BUS.

**HUB/SPOKE** Posee un canal de mensajes (*Message Broker*) centralizado, mientras que los SPOKE que son los adaptadores que convierten los mensajes en formato legible para el *Message Broker*, se encuentra distribuidos.

**BUS** Utiliza un soporte central de mensajes (*backbone*) que permite la propagación de mensajes. Los adaptadores en este caso se encuentran en la misma plataforma que la aplicación y requieren suscribir a la aplicación con el Bus.

Uno de los aspectos que destacan en esta tecnología, es que tanto protocolos como plataforma suelen ser propietarios, por lo tanto son dependientes del proveedor de la solución de integración. Por lo que la adición de funcionalidades y la comunicación con otros protocolos suelen ser características adicionales que se deben adquirir en lugar de desarrollar.

Como una evolución de las tecnologías de integración EIA al adoptar algunas características disponibles en SOA, aparece uno de los exponentes de integración en estos tiempos como son los buses de servicios empresariales.

#### 2.6.1.3. Bus de Servicios Empresarial

Definir un Bus de Servicios es una tarea algo complicada dado que no existe un consenso formal sobre qué características son las que deben existir en un Bus de Servicios. Estas características pueden variar según la visión que tenga el proveedor de esta solución para resolver el problema de integración. Sin embargo, basado en las características principales, según la definición del autor Menge en [49] se puede definir que un Bus de Servicios empresarial es “Una infraestructura de integración distribuida, de estándares abiertos y basado en mensajes que

proporciona servicios de enrutamiento, invocación y de mediación para facilitar las interacciones de las aplicaciones y servicios de manera segura y confiable". Un ESB reúne muchas de las características beneficiosas de las tecnologías anteriores dentro de SOA y EIA como bajo acoplamiento, integración distribuida y gran capacidad de escalamiento. Todo esto por medio de estándares y protocolos libres, Servicios Web, transformaciones y ruteo inteligente, lo que lo convierte en una tecnología muy estimada frente a necesidades de integración.

En el ámbito empresarial los ESB se destacan por cumplir algunos requisitos como los mencionados en[81]:

**Invocaciones con soporte multiprotocolo:** Puede realizar llamadas y recibir respuestas, para los cuales debe soportar estándares de Servicios Web como SOAP, WSDL, UDDI. Protocolos de transporte como HTTP, UDP, SSL. Comunicación con JMS<sup>23</sup>, entre otros.

**Enrutamiento basado en contenido:** Esto se refiere a la capacidad de determinar el destino de un mensaje. En ciertos casos es requerido analizar adicionalmente el contenido o seguir políticas para determinar la ruta adecuada.

**Transparencia de ubicaciones:** Los clientes requieren conocer la existencia del servicio más no como encontrar la ruta hacia él. El ESB recibe la petición y localiza el servicio.

**Calidad y seguridad:** Permitir utilizar tecnologías con capacidad de entrega garantizada como JMS. Permitir encriptamiento, desencriptamiento, autenticación.

**Patrones de intercambio de mensajes :** Soportar los mensajes petición/respuesta usados comúnmente en Servicios Web y otros como de publicación/-suscripción para Servicios de tiempo prolongado.

**Transformación:** Debe ser capaz de realizar las trasformaciones o traducciones necesarias sobre los mensajes para que la comunicación entre servicios o recursos dispares pueda ser llevado a cabo.

---

<sup>23</sup>Java Message Service: En una Api que permite a las aplicaciones crear, enviar y leer mensajes utilizando una comunicación fiable, asincrona y con bajo acoplamiento

## 2.6.2. Apache ServiceMix como Framework de Desarrollo

Apache ServiceMix dentro de la presente tesis ha sido seleccionado como base para la implementación de la plataforma de anotación semántica de servicios, ya que al ser un Bus de Servicios Empresarial contiene muchas de las características beneficiosas tratadas en la sección 2.6.1.3, además de otras mas específicas que han sido de especial interés en el desarrollo del proyecto, como:

- Proveer una tecnología de vanguardia principalmente para el manejo de Servicios Web.
- Brindar una estructura modular que permite la creación de rutas fácilmente configurables, y adaptables a las necesidades del entorno.
- Soportar de tecnologías Apache y Java, que ayudan al mantenimiento del proyecto gracias a la gran comunidad siempre activa detrás de los mismos.

Por esta razón, a continuación se describe en más detalle la herramienta Apache ServiceMix, su arquitectura y características principales que han permitido ser considerada de entre otras herramientas (sección 3.2.1.1) como *framework* de desarrollo para el prototipo planteado.

### 2.6.2.1. Apache ServiceMix

Apache ServiceMix es un Bus de Servicios empresarial de código abierto que soporta una serie de componentes de integración basados en JBI<sup>24</sup>, de los cuales adquiere su funcionalidad como Apache ActiveMQ<sup>25</sup>, Camel<sup>26</sup>, CXF<sup>27</sup>, y Karaf<sup>28</sup>. Componentes con los que se pretende brindar una poderosa plataforma de integración enfocada en ser flexible, fiable y de alta conectividad, para el desarrollo de soluciones personalizadas de integración [66]. Actualmente es una solución bastante reconocida en el ámbito de integración debido a que aplica estándares como JBI y usa tecnologías Apache, lo que le brinda alta escalabilidad. Es soportado principalmente por la comunidad y existen empresas como IONA<sup>29</sup> technologies quienes brindan soporte y entrenamiento.

---

<sup>24</sup>Java Business Integration: Es una especificación desarrollada bajo la Java Community Process (JCP) que permite definir una arquitectura de integración mediante el uso de componentes con bajo acoplamiento.

<sup>25</sup><http://activemq.apache.org/>

<sup>26</sup><http://camel.apache.org/>

<sup>27</sup><http://cxf.apache.org/>

<sup>28</sup><http://karaf.apache.org/>

<sup>29</sup>[www.iona.com](http://www.iona.com)

### 2.6.2.2. Arquitectura de Apache ServiceMix

Apache Servicemix está basado en los estándares JBI, por lo que comparte mucha de su arquitectura. Esto significa que para elaborar el producto de integración, se utiliza elementos llamados componentes que tienen la capacidad de ser enchufados entre sí dentro de un entorno JBI. En el caso de la arquitectura de Service Mix específicamente, dichos componentes están desarrollados como elementos básicos y complementarios entre sí, dispuestos de manera que llegen a formar una verdadera herramienta de integración. Como se muestran en la figura 2.14 como base se encuentra el componente karaf, el cual es el encargado de servir como contenedor y soporte de OSGI en tiempo de ejecución. A este contenedor se le suman componentes *Normalized Message Router* (NMR), los cuales son los encargados de brindar las capacidades de transformación, ruteo, paso de mensajes, etc. Finalmente se posee los elementos de comunicación con las aplicaciones y de soporte para diferentes protocolos como HTTP, REST, JMI, entre otros.

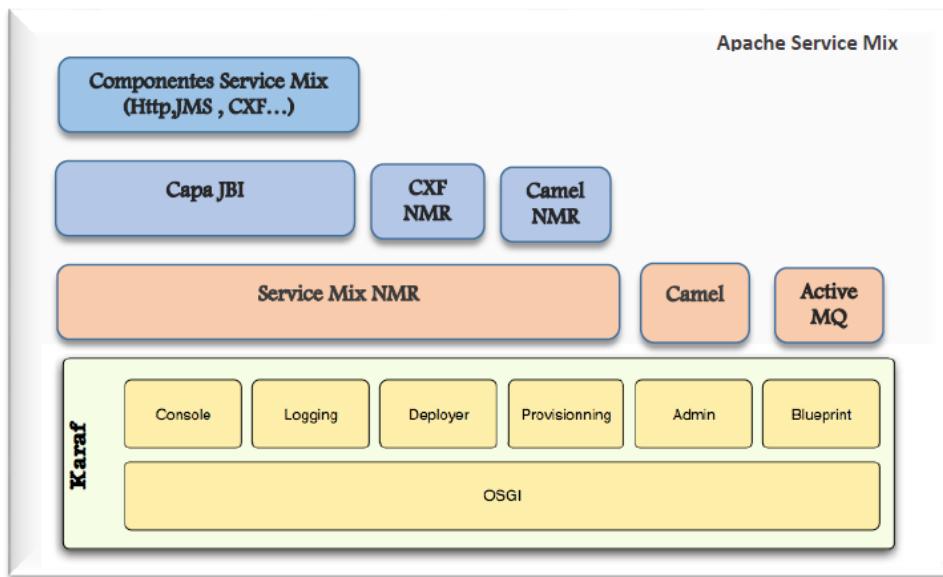


Figura 2.14: Arquitectura ServiceMix

A parte de los mencionados Apache ServiceMix hace uso de otros componentes tomados principalmente de diversos proyectos Apache (figura 2.15) como:

**Componente Camel :** Permite creación de rutas basados en algunos patrones de integración. Se pueden encontrar filtros, Splitter/Agregators, enriquecedores, resecuenciadores, etc.

**Componente CXF :** Permite comunicarse y soportar Servicios Web.

**Componente JMS (Active MQ):** Permite acceso a implementaciones JMS, entre el que se encuentran los basados en Active MQ<sup>30</sup>

**Componente HTTP:** Para acceder a SOAP y servicios basados en HTTP.

**Componente File :** Para poder acceder a archivos del sistema.

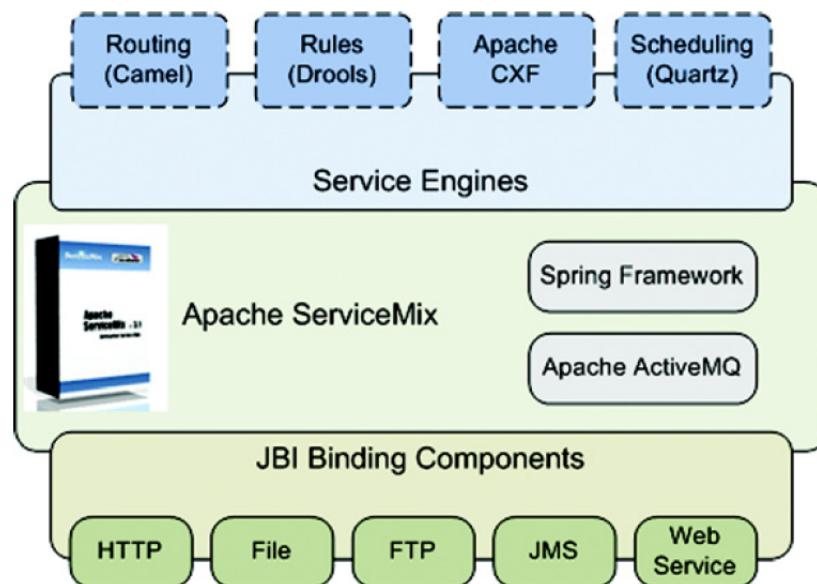


Figura 2.15: Vista de los componentes en Apache ServiceMix y su funcionalidad [66]

## 2.7. Patrones de integración para el diseño de la solución

Una de las características destacadas por lo que se ha seleccionado el Bus de Servicios Apache ServiceMix como *framework* de desarrollo como se indica en la sección 2.6.2.1, es por su capacidad de soportar de patrones de integración. Esta característica ha sido solicitada pues como se tratará a continuación sirve como medio de entendimiento común en el diseño y portabilidad de las soluciones dentro del sector de la integración.

<sup>30</sup>Apache ActiveMQ : Mensajería de código abierto popular y potente servidor de integración y patrones.

### 2.7.1. Antecedentes

En determinados ámbitos, existen algunos problemas que se repiten continuamente, frente a los que hay que plantear una solución. Esta solución puede variar según el criterio de quién lo plantea y el conocimiento que dispone en ese momento. Sin embargo con la experiencia se puede determinar cuál puede llegar ser la mejor solución para dicho problema en base a los resultados obtenidos. Dado que esta experiencia puede ayudar a quienes se enfrentan a situaciones similares, muchos autores sobre distintas áreas como Christopher Alexander en el área de la construcción con *The Timeless Way of Building* (1979), Ward Cunningham y Kent Beck en el ámbito de la industria del software Orientado a Objetos (1987), entre otros, analizan y proponen seguir un determinado patrón de solución, como recomendación, cuando un determinado problema se hace presente [82]. En el caso del área de integración no ha sido la excepción y es así como Gregor Hohpe and Bobby Woolf autores del libro *Enterprise Integration Patterns* (2004) proponen un conjunto de 65 patrones obtenidos en base a la experiencia para la integración de aplicaciones a nivel empresarial.

### 2.7.2. Patrones de integración

En la actualidad los patrones de integración empresarial han llegado a tener una importante popularidad en el ámbito de la integración, de manera que no se puede adentrarse en el mundo de la integración sin tener que tratar con dichos patrones. La popularidad que se les ha otorgado esta bien fundamentada ya que además de destacar por los procedimientos (patrones) que utilizan para enfrentarse a determinados problemas, brindan un medio de entendimiento común para los interesados en la materia, ya que cada uno de los patrones dispone de un nombre para identificarlos y una conceptuación visual de los elementos que representan a la solución. De esta manera se puede expresarse las necesidades de integración en función a los patrones y los desarrolladores de software de integración ofrecen muchos de estos patrones en forma de componentes para ser utilizados directamente frente a una necesidad de integración. Por ejemplo entre los componentes comunes nacidos de dichos patrones se pueden encontrar ruteo de mensajes dinámico o filtros. Se puede llegar a ver patrones de integración en la mayoría de ESB como Apache ServiceMix, JBOSS FUSE, MULE, etc.

### 2.7.3. Características de los Patrones de Integración

Para el planteamiento de los patrones, los autores Gregor Hohpe y Bobby Woolf basaron sus diseños y soluciones en un concepto de mensajería asíncrona, planteándose como la mejor vía para la implementación de soluciones de integración. Esto se debe a que este tipo de mensajes permiten según proponen en su libro [36] :

- Envío de mensajes con mayor fiabilidad.
- Mejora en el aprovechamiento de recursos por parte del solicitante y del receptor.
- Funcionamiento sobre múltiples plataformas/lenguajes.
- Comunicación de aplicación a aplicación de forma remota.
- Prevención de la sobrecarga del receptor.
- Administración de hilos por parte de la aplicación.
- Mediación entre aplicaciones

Los mensajes pueden llegar a ser desde simples *String*, *Arrays* o hasta Registros u Objetos, los mismos que están conformados por dos elementos, la cabecera y el cuerpo. La cabecera tiene datos más orientados hacia el control y descripción de metadatos. El cuerpo por otro lado, contiene la información principalmente para el destinatario.

Los patrones de integración más básicos se encuentran listados en la tabla 2.5, mientras patrones de integración adicionales se pueden encontrar en la figura 2.16 :

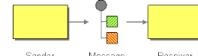
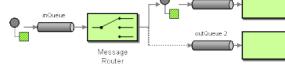
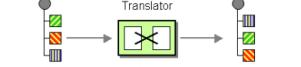
Patrón	Descripción	Representación Visual
Canal de Mensaje	Comunica aplicaciones, donde la primera escribe información en un extremo del canal y la segunda en otro extremo la lee.	
Mensaje	Conjunto de datos que se puede transmitir sobre un canal de mensajes.	
Tuberías y Filtros	Secuencias de pasos independientes que realizan una determinada tarea de procesamiento (filtro) que se comunican entre sí por canales de mensajes (tuberías)	
Router	Redirige los mensajes sobre diferentes canales dependiendo de ciertas condiciones.	
Traductor	Permite traducir de un formato a otro dependiendo de la necesidad.	
Endpoint	Cliente del sistema de mensajería que la aplicación puede usar para enviar y recibir mensajes.	

Tabla 2.5: Patrones base

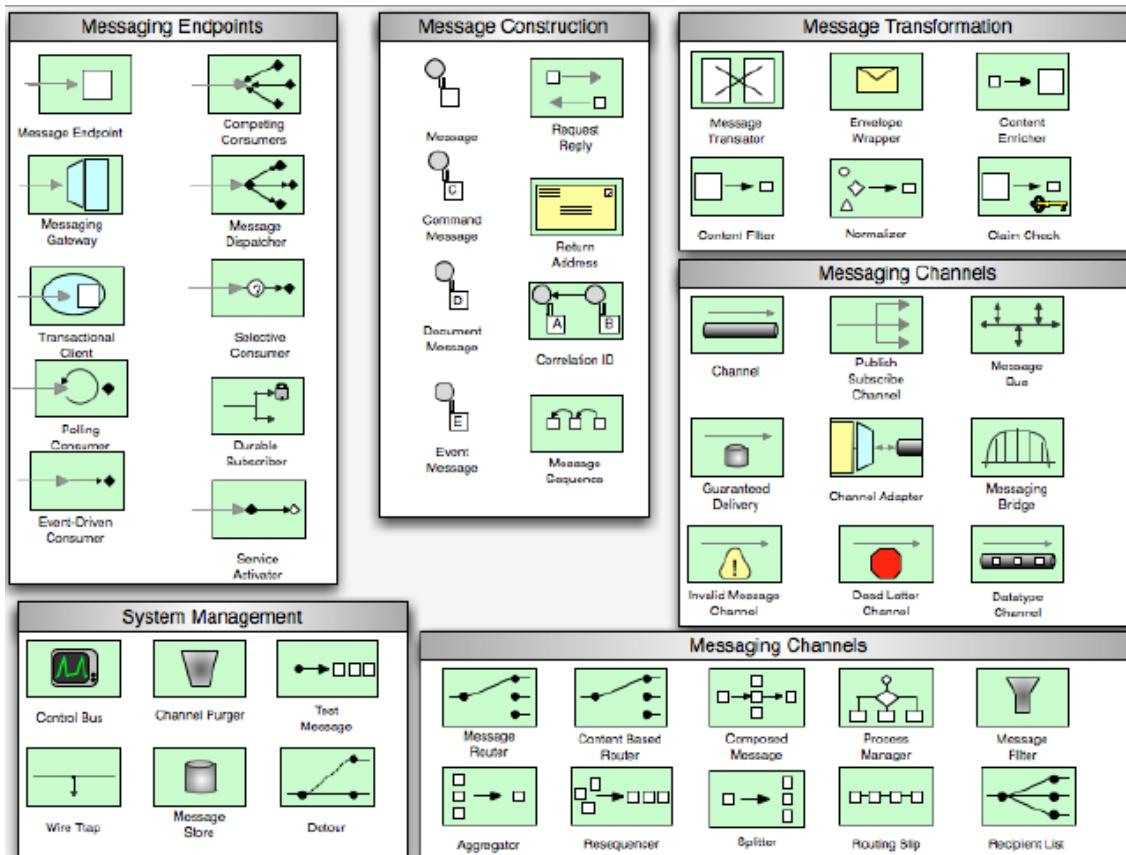


Figura 2.16: Varios patrones de integración [36]

## **Capítulo 3**

---

### **Proceso de Anotación Semántica**

---

Como se mencionó en el Capítulo 1 la presente tesis busca ampliar el enfoque de anotación semántica propuesto en [67], cubriendo dos aspectos necesarios para su mejoramiento, estos son: Brindar una implementación de software que de soporte a este enfoque de anotación y adicionar una etapa de selección automática de ontologías. En el presente capítulo se cubre el primero de ellos, exponiendo el proceso que se siguió para crear una implementación de este enfoque de anotación.

La implementación del proceso de anotación definido en [67] sobre una plataforma permite continuar la investigación en temas de anotación semántica de Servicios Web. Esto debido a que los autores de [67] solamente crearon prototipos simples (Experimentos específicos escritos sobre código Java), los cuales fueron suficientes para probar la validez del enfoque pero carecen de flexibilidad puesto que no permiten ejecutar nuevos experimentos ni modificar los algoritmos definidos.

Es por esta razón que este capítulo se cubre el diseño y creación de una implementación para este enfoque sobre una plataforma: con integración a Servicios Web, modular y estandarizada denominada Bus de Servicios. Este capítulo está dividido en dos secciones: la primera es el modelamiento, donde, se explica el enfoque de anotación a más detalle y se diseña la plataforma con sus componentes; la segunda es la implementación, en la que se dan detalles más específicos y técnicos del funcionamiento de cada componente definido en el diseño.

### 3.1. Modelamiento del proceso de anotación de Servicios Web

El enfoque de anotación definido en [67] es presentado de forma general, sin entrar en detalles específicos. Sin embargo, la implementación requiere una especificación más detallada del proceso de anotación. Por esto se dividió el modelamiento en dos partes: Primero, explicación del enfoque de anotación propuesto en [67]. Segundo, modelamiento planteado en esta tesis para la implementación de este enfoque de anotación sobre un Bus de Servicios.

#### 3.1.1. Adopción del proceso de Anotación Semántica

El sistema de anotación semántica propuesto en [67] consta de tres componentes principales: 1) Descripción sintáctica, 2) Proceso de anotación semántica y 3) Repositorio. En la figura 2.9 se muestra como interoperan estos componentes.

Los Servicios Web que son proporcionados manualmente por un usuario o bien tomados desde Internet (API, Aplicación Web) son ingresados al sistema, a lo cual los componentes de descripción sintáctica, anotación y repositorio comienzan a actuar. A continuación se describe estos componentes:

*El componente de descripción sintáctica:* se encarga de la invocación automática del servicio y la extracción de los parámetros (entradas, salidas) del Servicio Web. Para la gestión de la información los autores plantearon la utilización de un modelo de objetos, el cual se muestra en la figura 3.1. Este modelo de objetos además de los parámetros del Servicio Web considera las instancias obtenidas en las invocaciones del servicio.

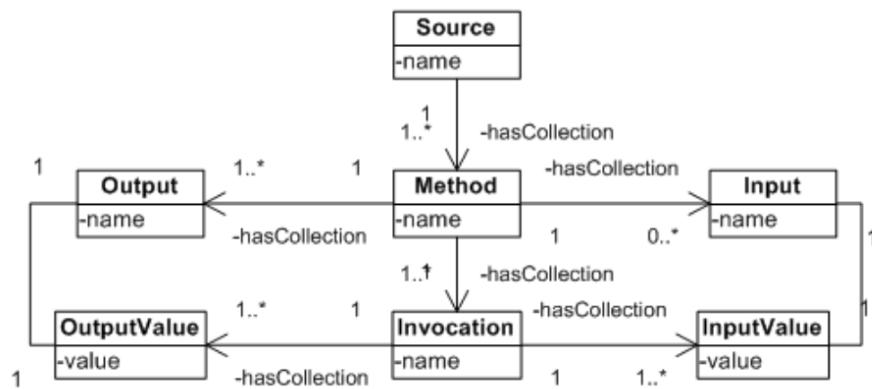


Figura 3.1: Modelo de objetos para la descripción sintáctica de un Servicio Web

*Proceso de anotación semántica:* Es el componente principal dentro del sistema de anotación, este incluye: la definición de un proceso de enriquecimiento de la descripción sintáctica de los servicios mediante recursos externos (Sinónimos, Sugerencias), emparejamiento (*matching*) con los conceptos de la ontología usando medidas de similitud y finalmente mecanismos de validación de anotaciones obtenidas. La información generada por este proceso es almacenada dentro de un nuevo modelo de objetos (figura 3.2). Sin embargo, dicho modelo está especializado para información geográfica, debido a que incluye atributos de longitud y latitud.

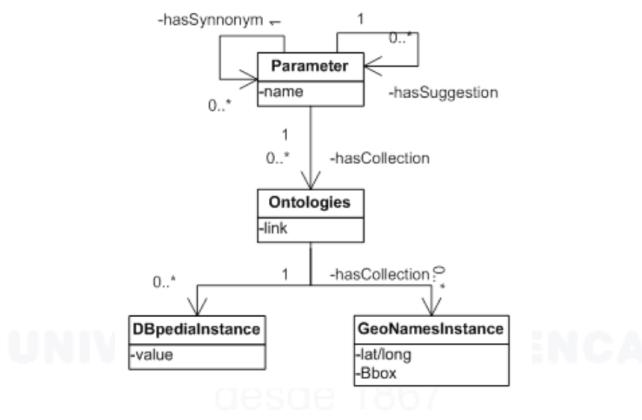


Figura 3.2: Modelo de objetos para descripción de anotaciones

*Repositorio:* Es en donde se almacenan los resultados obtenidos tanto por el componente de descripción sintáctica como por el proceso de anotación. Adicionalmente el sistema plantea la posibilidad de la generación automática de descripciones semánticas de los Servicios Web anotados en lenguajes como: hREST, SA-REST, MicroSWSA a partir de la información del repositorio.

### 3.1.2. Definición y adaptación de las etapas del proceso de anotación

En esta sección se define el diseño de la plataforma de anotación semántica de Servicios Web mejorada basada en [67], en la cual se cubre tres aspectos de diseño: arquitectura general, modelamiento del proceso de anotación y adaptación del modelo de objetos. Como se mencionó en el capítulo 1, se planteó la utilización de un Bus de Servicios como plataforma de desarrollo para el proceso de anotación semántica. Sin embargo, adicionalmente al Bus de Servicios, se requieren otros componentes de software que completen el sistema. Es así que se plantearon dos aplicaciones suplementarias al Bus de Servicios, estas son: una interfaz gráfica que

facilite la utilización del sistema de anotación y un Servicio Web que proporcione la funcionalidad de **selección de ontologías**.

*Arquitectura general del sistema:* En la figura 3.3 se muestra la arquitectura planteada en este trabajo, en esta propuesta de arquitectura se considera las siguientes adaptaciones a lo definido en [67]: los componentes originales de descripción sintáctica y anotación semántica son divididos en más componentes (detallados en la sección 3.2.2) e implementados dentro del Bus de Servicios; el repositorio es gestionado a través de una base de datos relacional externa al Bus de Servicios; se adiciona una interfaz gráfica (Web) que se encarga de la interacción con el usuario; Finalmente, los recursos externos están formados por Servicios Web.

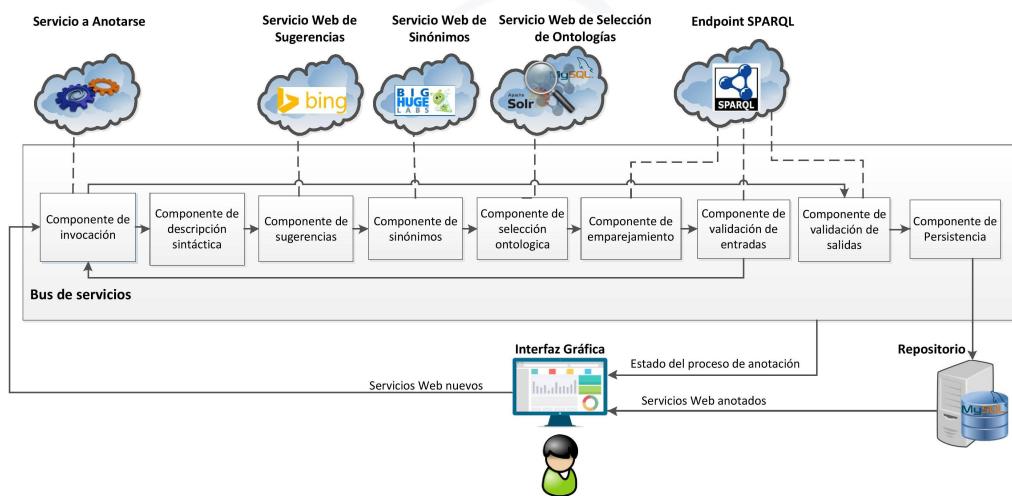


Figura 3.3: Arquitectura general de la plataforma planteada

*Proceso de anotación:* Para el desarrollo del proceso de anotación dentro del Bus de Servicios se optó por la utilización de patrones de integración [36], en los cuales se establece: transformaciones, eventos, ruteo, mensajes, filtros y otras herramientas (ver sección 2.7) que permiten generar soluciones de integración complejas. Estas características facilitaron el modelamiento e implementación del proceso de anotación. Adicionalmente, los patrones de integración poseen una representación gráfica que permite la creación de diagramas, a través de los cuales se puede ver la lógica que sigue el flujo de información entre los componentes creados.

El proceso de anotación semántica de Servicios Web se modeló utilizando la representación gráfica de los patrones de integración, la lógica que sigue este proceso fue tomada de [67] y se muestra en la figura 3.4. Utilizando este flujo se

implementó el proceso de anotación dentro del Bus de Servicios. En la sección 3.2.2 se describe con más detalle cada uno de los componentes e interacciones entre componentes planteadas, también se proveen detalles de implementación.

*Modelo de objetos:* Respecto al modelo de objetos que se utiliza durante todo el proceso de anotación y que es almacenado en el repositorio, ha sido ajustado levemente con respecto al original. Este nuevo modelo planteado se muestra en la figura 3.6

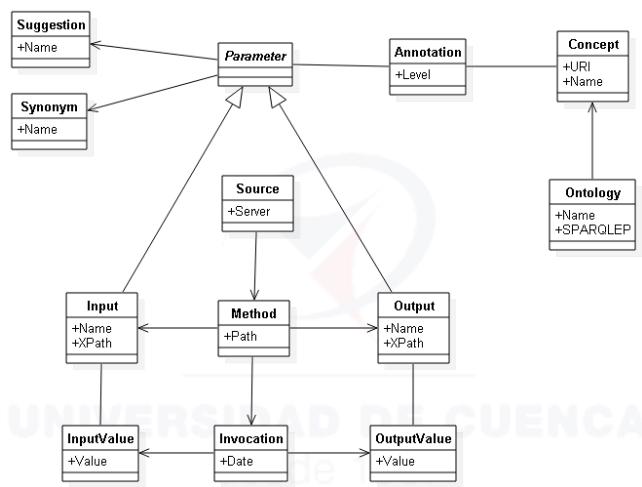


Figura 3.6: Modelo de objetos propuesto

Los principales cambios realizados al modelo original son:

- Se unificaron los diagramas 3.2 y 3.1.
- Se eliminaron las clases específicas de dominio Geográfico (*GeoNamesInstance*, *DBpediaInstance*).
- Se aplicó una relación de herencia entre *Input* y *Output* con la clase abstracta *Parameter*.
- Las relaciones *hasSynonym* y *hasSuggestion* se sustituyeron por las clases *Suggestion* y *Synonym*.
- Se reemplazó la clase *Ontologies* por *Concept*.
- La relación (anotación) entre *Parameter* y *Ontologies* fue reemplazada por una relación de clase enlace (*Annotation*).
- Se agregó una representación para las ontologías con la clase *Ontology*.

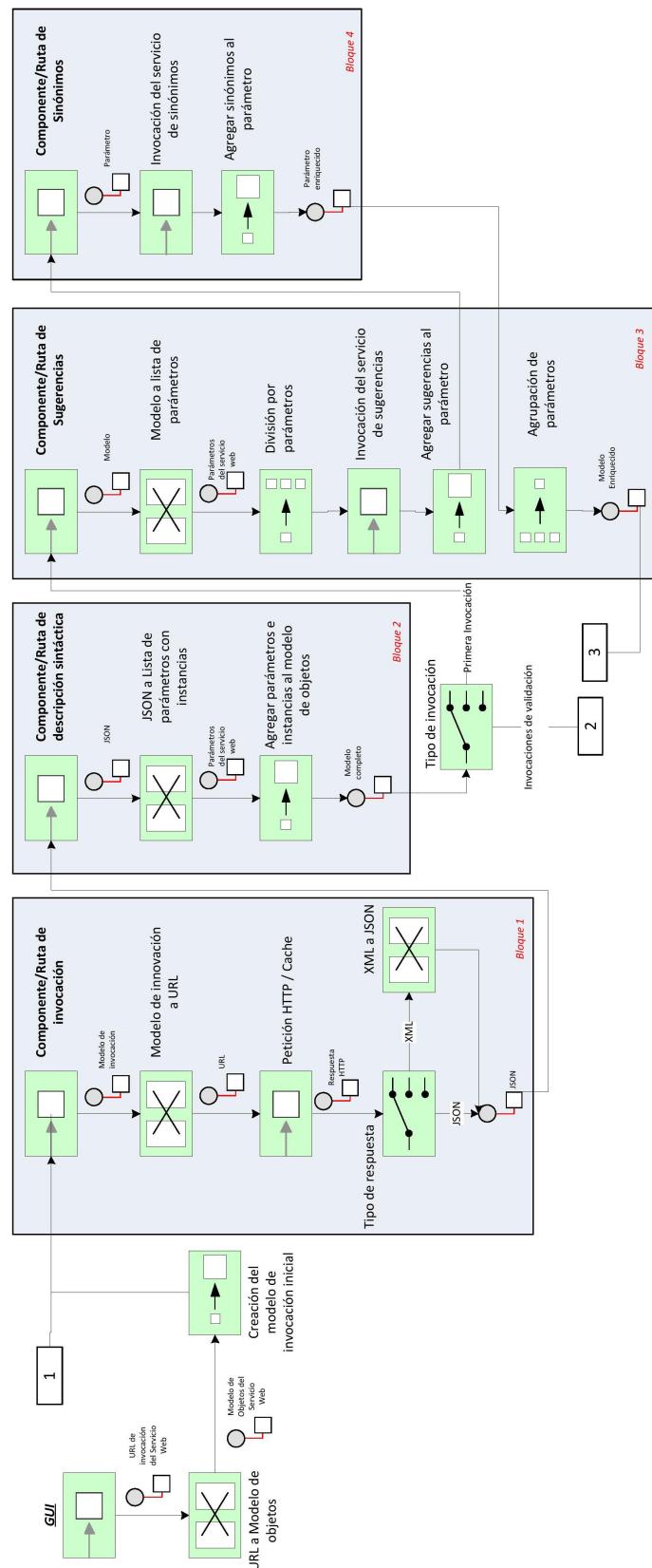


Figura 3.4: (1/2)Proceso de anotación expresado con patrones de integración.

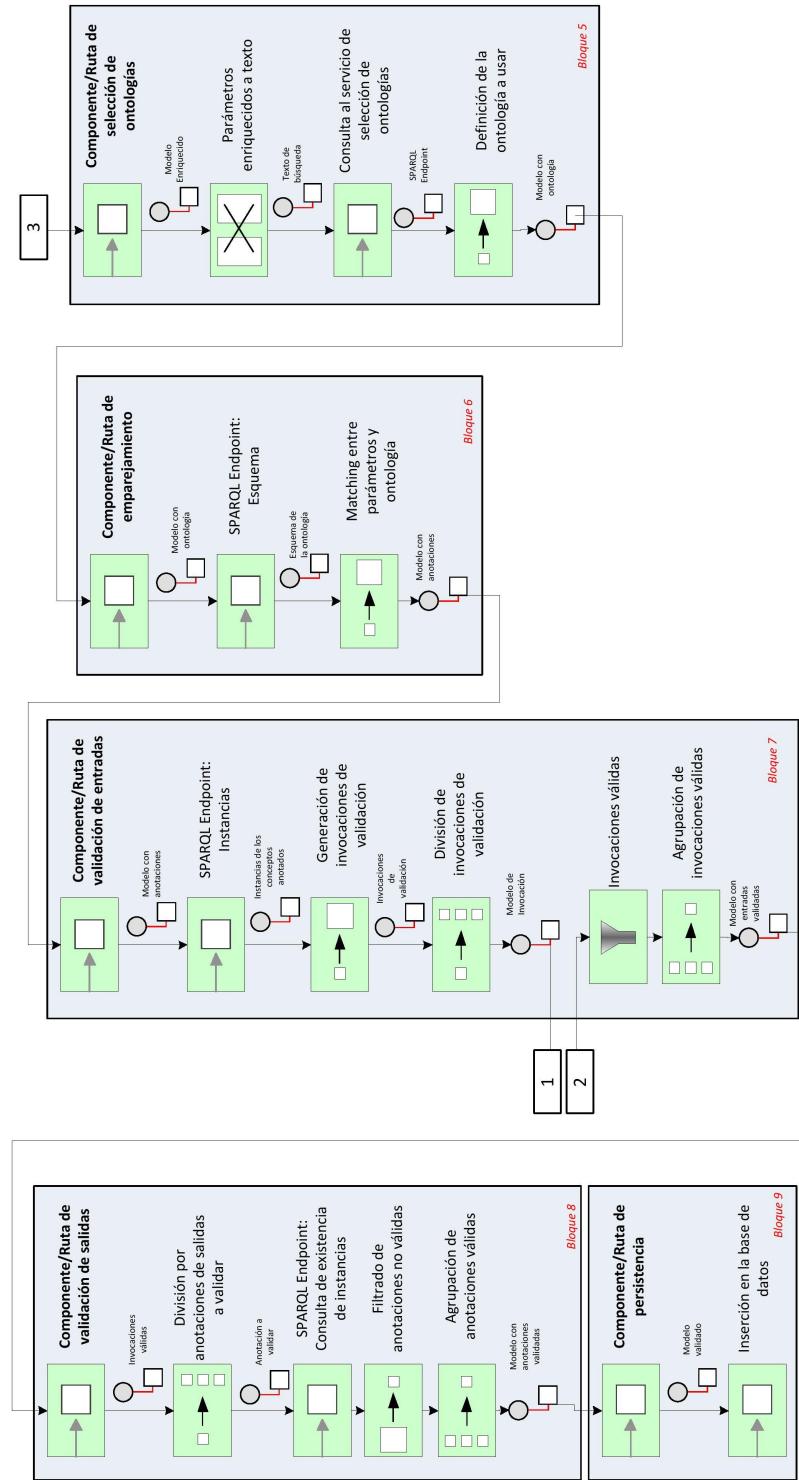


Figura 3.5: (2/2) Proceso de anotación expresado con patrones de integración.

Los cambios al modelo original se realizaron debido a las necesidades prácticas (de implementación) del proceso de anotación respecto al planteamiento teórico original. Este nuevo modelo es utilizado como formato de mensaje entre todas las etapas del proceso, es decir este modelo fluirá y se enriquecerá entre las distintas etapas del proceso de anotación. Finalmente, este modelo una vez tenga toda la información de las anotaciones será almacenado en el repositorio a través de un modelo relacional (ER) equivalente.

## 3.2. Implementación del proceso de anotación sobre un Bus de Servicios

En esta sección se describe la implementación del proceso de anotación semántica, siguiendo los modelos y diagramas creados.

### 3.2.1. Análisis y selección de las herramientas

Para la implementación de la plataforma de anotación semántica planteada se requieren esencialmente tres tipos de herramientas base: Bus de Servicios, Base de Datos, Servidor de Aplicaciones. A continuación se detallan las necesidades que deben cubrir cada tipo de herramienta, el análisis y selección de software a ser usado.

#### 3.2.1.1. Bus de Servicios

Es la herramienta principal requerida dentro de la plataforma, debido a que sobre esta se implementa todo el proceso de anotación, es decir, sobre esta herramienta se implementa lo establecido en el diagrama 3.4. Por lo considerado en el diseño, la herramienta de software seleccionada debería soportar: patrones de integración, desarrollo por componentes, interoperabilidad con Servicios Web y programación orientada a objetos.

Actualmente existe un gran número de Buses de Servicio OpenSource disponibles (JBoss Fuse, ServiceMix, MuleESB, etc. Sección 2.6.1.3) y cada uno tiene una forma diferente de abordar el problema de la integración de aplicaciones. Sin embargo, solo los buses de servicios basados en ServiceMix (Apache ServiceMix y JBoss Fuse) cumplen con los requerimientos establecidos: Implementan los patrones de integración a través de su componente Apache Camel, su desarrollo

está basado en el uso de componentes a través de OSGi<sup>1</sup> (Apache Karaf), poseen conectores que permiten el consumo de Servicios Web (Apache CXF, Apache HTTP Client) y están escritos en el lenguaje de programación Java que permite POO<sup>2</sup>.

Entre las dos opciones disponibles (Apache ServiceMix y Jboss Fuse) se decidió utilizar ServiceMix por tres razones: primero, este Bus de Servicios cubre con todos los requerimientos para el desarrollo del sistema planteado; segundo, JBoss Fuse al ser un software profesional tiene componentes especializados como Fabric8 que lo vuelven más pesado (mayores requerimientos de hardware) y que no se necesitan para este desarrollo; tercero, la comunidad de desarrolladores de Apache ServiceMix es más activa y abierta, Jboss Fuse al tener soporte propietario no posee foros abiertos para desarrolladores independientes. Además, al tener ambos Buses de Servicios el mismo núcleo, la migración de ServiceMix a Fuse sería rápida.

### 3.2.1.2. Base de datos

Esta herramienta permite el almacenamiento de los resultados generados por el proceso de anotación. Según se indicó en las especificaciones de diseño esta base de datos es relacional y debe ser fácilmente consultada tanto desde el Bus de Servicios como desde las Aplicaciones Web.

Actualmente existe una gran variedad de bases de datos relacionales dentro del ambiente OpenSource. Entre las bases de datos más utilizadas tenemos: MySQL<sup>3</sup>, PostgreSQL<sup>4</sup>, SQLite<sup>5</sup>, todas estas opciones tienen conectores que permiten usarlas desde distintos lenguajes de programación y plataformas. Sin embargo, la más popular de entre todas ellas es MySQL, la cual tiene una comunidad muy activa de desarrolladores, amplia documentación, software adicional, etc.

Se decidió utilizar MySQL como la base de datos principal de toda la plataforma por dos razones: primero, como se mencionó MySQL es la base de datos más popular actualmente, esto implica que la documentación y soporte en foros es abundante; segundo, como se explica a más detalle en la sección 2.5.3 esta base de datos permite realizar búsquedas basadas en texto, esta funcionalidad se requiere para la etapa de selección de ontologías (Ver sección 4.4.2.1).

---

<sup>1</sup><http://www.osgi.org/>

<sup>2</sup>Object-oriented programming

<sup>3</sup><https://www.mysql.com/>

<sup>4</sup><http://www.postgresql.org/>

<sup>5</sup><https://www.sqlite.org/>

### 3.2.1.3. Servidor de Aplicaciones Web

Adicionalmente al desarrollo del proceso de anotación semántica se planteó el desarrollo de otros dos componentes de software dentro de la plataforma. Primero, un Servicio Web el cual permita el acceso a las funciones de selección automática de ontologías. Segundo, una interfaz gráfica Web desde la cual se puede acceder a las funcionalidades del sistema de manera amigable para el usuario. Ambos componentes necesitan un Servidor de Aplicación sobre el cual desplegarse.

Debido a que todas las tecnologías usadas en este proyecto están basadas en la plataforma Java se decidió utilizar un servidor de aplicaciones que siga este lineamiento. Los principales servidores de aplicaciones Open Source basados en Java son: Glassfish<sup>6</sup>, Jboss<sup>7</sup> y Tomcat<sup>8</sup>. Estas servidores soportan: JSP<sup>9</sup>, JSF<sup>10</sup>, JPA<sup>11</sup> entre otras tecnologías Web. Sin embargo, los dos primeros son usados principalmente para ambientes de producción, además tienen soporte comercial y requieren más recursos de hardware. Tomcat por su parte es un servidor más ligero, que tiene soporte a través de su comunidad de desarrolladores, haciéndolo ideal para el desarrollo de prototipos.

Se decidió utilizar Tomcat por ser la más simple y liviana de entre las opciones, pero que, sin embargo, posee las funcionalidades requeridas para el desarrollo de este sistema. Las tecnologías que se utilizaron para el desarrollo de las dos aplicaciones son: JSP para el Servicio Web de selección de ontologías y JPA con JSF (PrimeFaces<sup>12</sup>) para la interfaz gráfica.

### 3.2.2. Descripción de componentes

A continuación se describe el funcionamiento de cada uno de los componentes planteados dentro del proceso de anotación, adicionalmente, en cada componente se explican los detalles de implementación más relevantes.

#### 3.2.2.1. Componente de invocación

Este componente se encarga de invocar al Servicio Web que se pretende anotar. Las invocaciones dentro del proceso de anotación pueden ser de dos tipos:

---

<sup>6</sup><https://glassfish.java.net/>

<sup>7</sup><http://jbossas.jboss.org/>

<sup>8</sup><http://tomcat.apache.org/>

<sup>9</sup><http://www.oracle.com/technetwork/java/javaee/jsp>

<sup>10</sup><http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>

<sup>11</sup><http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>

<sup>12</sup><http://primefaces.org/>

primero, las invocaciones iniciales que son las que se ejecutan cuando el proceso de anotación inicia. Estas invocaciones permiten obtener una respuesta de ejemplo del Servicio Web, con la cual se extrae una descripción sintáctica (entradas y salidas) del servicio; segundo, las invocaciones de validación, que son generadas por el componente de validación de entradas, que permiten verificar las anotaciones realizadas sobre las entradas del servicio.

Este componente recibe como entrada una instancia del modelo de objetos 3.6 del Servicio Web con los valores de entradas a ser utilizados para la llamada al servicio. Estos valores en conjunto con la información sintáctica que se encuentra en el modelo de objetos (servidor, método y nombre de parámetros) son transformados en una URL, la cual es invocada.

Respecto a las repuestas de las invocaciones, en el presente trabajo se planteó usar exclusivamente Servicios Web que trabajen con los formatos JSON y XML, debido a que son los formatos más utilizados. Esto implica que los servicios que trabajen con formatos distintos serán considerados como no invocables. La etapa final de este componente consiste en estandarizar la respuesta o salida, para esto se definió un lenguaje único al cual las respuestas son transformadas, esto con el fin de simplificar las etapas posteriores del proceso. En lenguaje seleccionado fue JSON, es decir, si las repuestas están en formato XML son transformadas a JSON. Un ejemplo de la ejecución de este componente se muestra en la figura 3.7.

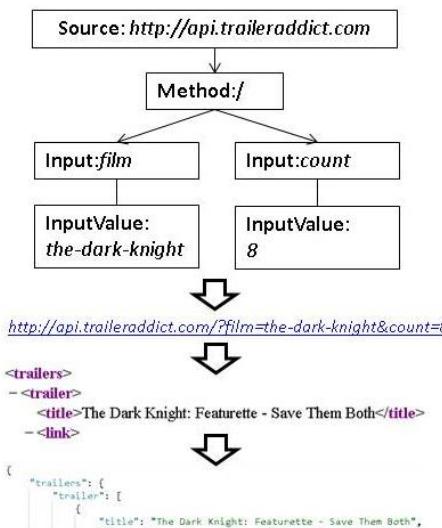


Figura 3.7: Ejemplo de ejecución del componente de invocación

## Consideraciones de implementación

- Para facilitar el manejo de los servicios dentro del componente, solamente se soportan las URLs estándar de tipo:  
`http://server/method?param1=value1&param2=value2...`
- Para evitar la repetición de invocaciones a los Servicios Web se implementó una caché usando las URLs de los servicios invocados como clave.

### 3.2.2.2. Componente para la descripción sintáctica

Este componente se encarga de la extracción de parámetros (con sus instancias) de un Servicio Web a partir de la respuesta obtenida (JSON) de la invocación. Esta funcionalidad es requerida en dos etapas del proceso de anotación: en la invocación inicial al Servicio Web se utiliza para completar el modelo de objetos (3.6), con los parámetros de salida del Servicio Web y en las invocaciones posteriores, permite agregar nueva información al modelo de objetos sobre nuevas instancias (valores de los parámetros).

El componente recibe un documento JSON proveniente del componente de invocación y lo transforma en una lista de parámetros simple. Este proceso de transformación tiene dos etapas principales: primero, el documento JSON es analizado y transformado en una estructura de árbol (Objeto en memoria); segundo, esta estructura es recorrida recursivamente con el fin de detectar solamente los elementos terminales (Hojas), debido a que estos elementos son los que contienen los valores (instancias). A continuación, a cada parámetro detectado se le asigna una lista de instancias, la cual consiste de todos los valores correspondientes para dicho parámetro encontrados dentro de la respuesta del servicio.

Con el objetivo de facilitar futuros análisis sobre los parámetros encontrados se conservó su ruta interna dentro de la respuesta del Servicio Web, es decir, se almacenó el recorrido que debe realizarse para llegar desde la raíz del documento hasta el elemento terminal (parámetro). El formato usado para guardar esta ruta se basa en el estándar XPath<sup>13</sup>, y se almacena en el atributo *XPath* de las clases *Input* y *Output*. Un ejemplo del proceso de extracción de parámetros de un Servicio Web se muestra en la figura 3.8.

---

<sup>13</sup><http://www.w3schools.com/xpath/>

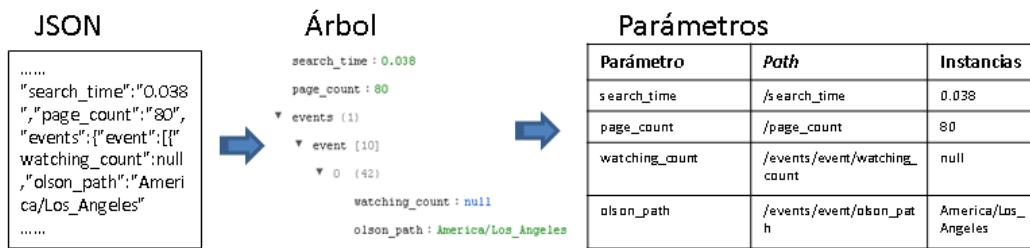


Figura 3.8: Ejemplo de extracción de parámetros

El resultado final de este componente es el modelo de objetos con el nombre de parámetros de salida y sus instancias (o valores). Este modelo puede ser utilizado para la descripción sintáctica del servicio (primera invocación) o bien para la etapa de validación (validación de entradas)

### Consideraciones de implementación

- Se utilizó la librería Jackson<sup>14</sup> para el análisis de los documentos JSON (*parsing*), debido a que este se encuentra disponible mediante un componente (*Bundle*) dentro de ServiceMix.

#### 3.2.2.3. Componente de enriquecimiento de términos (Sugerencias, Sinónimos)

Este componente se encarga de agregar “alias” o términos equivalentes al nombre de cada uno de los parámetros obtenidos en la descripción sintáctica del Servicio Web. Estos nuevos términos se usan posteriormente en la etapa de emparejamiento semántico para mejorar el número de anotaciones con las ontologías. Existen dos tipos de enriquecimiento que se han considerado para este componente: sugerencias y sinónimos, esto debido a que solucionan los problemas más frecuentes en la etapa de *matching*. Por ejemplo al agregar sugerencias a los parámetros de los servicios se evitan posibles errores (ortográficos), por lo que un parámetro llamado “sise” puede ser relacionado con “size” mediante el uso de sugerencias. Segundo permite la separación de palabras que estén dentro de términos, es decir la separación de nombres compuestos de parámetros. Por ejemplo un parámetro “totalsize” será separado en “total” y “size”. Por otro lado, el agregar sinónimos abre la posibilidad de que un parámetro se anote con ciertos conceptos aunque no tengan exactamente el mismo nombre. Por ejemplo

<sup>14</sup><https://github.com/FasterXML/jackson>

si un parámetro se llama “street” y existe un concepto llamado “road”, el uso de sinónimos puede permitir la anotación de este parámetro.

Para la implementación del este componente se combinó estos dos tipos de enriquecimientos (siguiendo lo establecido en [67]) como se exemplifica en la figura 3.9. El enriquecimiento con sinónimos se realiza a partir de las sugerencias encontradas, es decir, para cada sugerencia de los parámetros se buscan sinónimos.

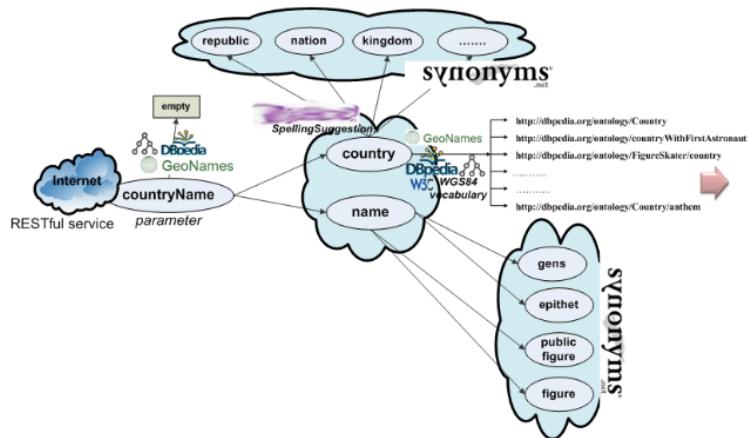


Figura 3.9: Enriquecimiento por sinónimos y sugerencias [67]

Para la implementación de estas dos etapas (sinónimos y sugerencias) se decidió separar en dos componentes (rutas) dentro del Bus de Servicios, de esta manera el enriquecimiento con sinónimos podría ser utilizado independientemente al de sugerencias (figura 3.4, bloques 3 y 4). Esto permite evaluar las anotaciones independientemente y detectar la influencia de los sinónimos y sugerencias.

El resultado final de este componente es el modelo de objetos enriquecido con sugerencias y sinónimos para cada uno de los parámetros que posea el Servicio Web, este modelo enriquecido es usado en fases posteriores del proceso de anotación semántica.

### Consideraciones de implementación

- El servicio de sugerencias utilizado: Bing Search<sup>15</sup> de Microsoft.
- El servicio de sinónimos utilizado: Big Huge Thesaurus<sup>16</sup>.
- Al igual que en el componente de invocación se implementó una caché para evitar alcanzar los límites de invocación de los servicios.

<sup>15</sup><http://datamarket.azure.com/dataset/bing/search>

<sup>16</sup><https://words.bighugelabs.com/>

- Debido a que el servicio de sinónimos provee demasiadas respuestas(más de 10) por consulta se decidió considerar solamente los 3 primeros sinónimos retornados, que a su vez son los que poseen mayor relación con la palabra consultada.

#### 3.2.2.4. Componente de selección de ontologías

Este componente selecciona automáticamente la ontología más representativa (permite el mayor número de anotaciones) para el Servicio Web. Este componente recibe el modelo enriquecido y a partir de este encuentra una ontología con un SPARQL Endpoint activo que pueda ser usado en las etapas posteriores del proceso de anotación.

Con el objetivo de simplificar el desarrollo de este componente dentro del Bus de Servicios se decidió crear un Servicio Web externo que provea la funcionalidad de selección de ontologías, de manera que el componente dentro del Bus de Servicios se limita a invocar a dicho servicio. Los detalles de análisis e implementación del servicio de selección de ontologías se describen en la sección 4.4.3.

Dentro del Bus de Servicios este componente transforma el modelo de objetos a una cadena de texto (tomando nombres de parámetros, sugerencias y sinónimos) y la envía al Servicio Web desarrollado, la respuesta del servicio es la URL del SPARQL Endpoint de la ontología seleccionada. El resultado final de este componente es la inclusión de la información de la ontología (seleccionada) en el modelo de objetos, la cual se utiliza en etapas posteriores del proceso de anotación. En la figura 3.10 se muestra un ejemplo de la ejecución de este componente.

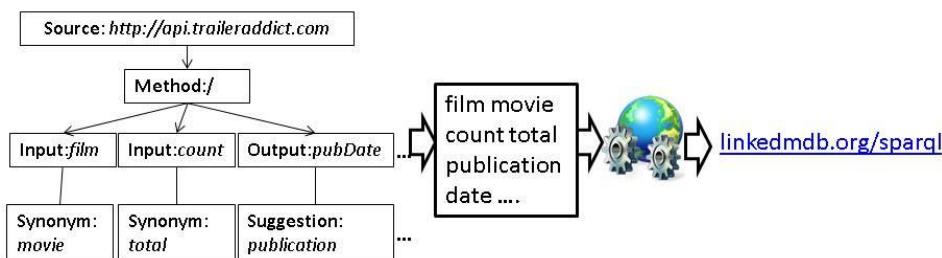


Figura 3.10: Ejemplo de ejecución del componente de selección de ontologías

#### Consideraciones de implementación

- Se planteó la posibilidad que el usuario seleccione por sí mismo una ontología, en este caso el servicio no es invocado, simplemente se asigna la ontología del usuario dentro del modelo y se continúa a la siguiente etapa.

- Para las peticiones al Servicio Web desarrollado se utilizó HTTP POST debido a que las descripciones sintácticas de ciertos servicios pueden superar las limitaciones de las peticiones HTTP GET, en cuanto a la cantidad de parámetros enviados.

### 3.2.2.5. Componente de *Matching* o emparejamiento

Este componente es el encargado de crear las anotaciones entre los parámetros (enriquecidos) del Servicio Web y los conceptos de la ontología seleccionada, es decir, para cada parámetro del Servicio se busca conceptos dentro de la ontología con los cuales guarde una relación (Sintáctica). La búsqueda de estos conceptos se realiza a través de una comparación netamente sintáctica (comparación de cadenas de texto) entre el nombre de los parámetros (incluyendo sinónimos y sugerencias) y el nombre de los conceptos (Clases y propiedades) de la ontología. Las anotaciones obtenidas en este componente son almacenadas dentro del modelo de objetos y validadas en la siguiente etapa del proceso.

En este componente se han definido tres etapas principales: 1) Obtención del esquema (Conceptos) de la ontología, 2) *Matching* o emparejamiento y 3) creación de las anotaciones dentro del modelo de objetos.

1. La obtención del esquema de la ontología se realiza mediante consultas al SPARQL Endpoint de la ontología seleccionada. Las consultas usadas para descargar el esquema de una ontología se describen a más detalle en la sección 4.4.2.3.
2. La etapa de *Matching* entre conceptos y parámetros consiste en la comparación de los nombres (incluyendo sinónimos y sugerencias) de los parámetros con el nombre de los conceptos (entidades y propiedades) dentro del esquema de la ontología. Para la realización de dicha comparación se utilizaron medidas de similitud en lugar de *Matchs* exactos, debido a que esto flexibiliza la comparación incrementando el número de anotaciones [67]. Se evaluaron varias medidas de similitud (promediando sus resultados) entre cada par: Concepto, Parámetro, y se utilizó un umbral para definir si son o no equivalentes. Los detalles específicos de la utilización de las medidas de similitud se explican en la sección 4.4.2.3. De esta manera se tiene un conjunto de relaciones: Concepto-Parámetro que son las posibles anotaciones del Servicio Web con respecto a la ontología seleccionada.

3. La creación de anotaciones dentro del modelo de objetos consiste en la creación de las relaciones encontradas (anotaciones Parámetro-Concepto) dentro de la clase *Annotation*. Una innovación que se planteó en el presente trabajo es agregar un estado a las anotaciones (Atributo *Level* de la clase *Annotation*), en el que se almacena el nivel que tuvo cada anotación dentro del proceso. Los niveles definidos para las anotaciones se detallan en la sección de validación de anotaciones 3.2.2.6.

El resultado final de este componente es el modelo de objetos completo donde se describa: Descripción sintáctica del servicio (Parámetros enriquecidos), Ontología seleccionada (Conceptos) y las anotaciones entre conceptos y parámetros. En la figura 3.11 se muestra un ejemplo de emparejamiento, donde las líneas de color azul-verde representan las anotaciones encontradas.

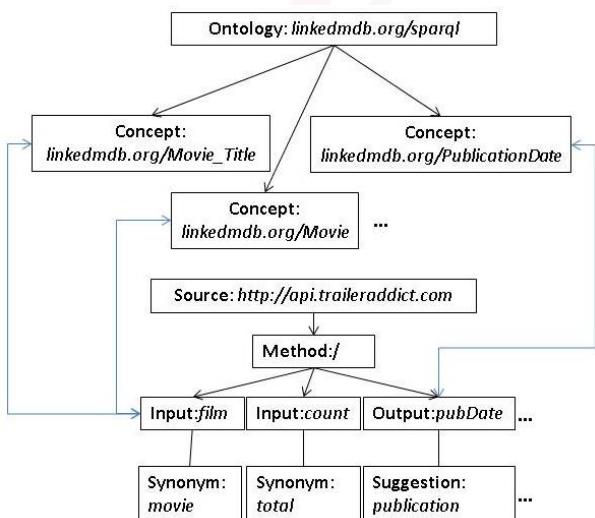


Figura 3.11: Ejemplo de ejecución del componente de emparejamiento

### Consideraciones de implementación

- Para facilitar la realización de pruebas del proceso de anotación, se decidió utilizar dos valores umbral diferentes en la etapa de emparejamiento, uno para las entradas y otro para las de salidas. Los mismos que son parametrizables y se reciben desde la interfaz gráfica al iniciar el proceso de anotación.
- Para la consulta al SPARQL Endpoint se utilizó la librería JENA.

- Para la aplicación de las medias de similitud se utilizó la librería Simmetrics<sup>17</sup>.
- Se adicionó un parámetro que defina el tiempo máximo de espera de las consultas al SPARQL Endpoint, este parámetro es recibido desde la interfaz gráfica.
- El usuario puede enviar opcionalmente una lista de parámetros a ignorar en la etapa de *Matching*, ya que ciertos parámetros no aportan información relevante al contexto del Servicio Web y deberían ser ignorados. Algunos ejemplos de estos parámetros son: contadores de página, tiempos de respuesta del servicio, formato de la respuesta, etc.

### 3.2.2.6. Componente de validación

Este componente verifica las anotaciones obtenidas en el componente de *Matching*, es el último paso definido dentro del proceso de anotación semántica planteado en [67]. El objetivo es validar que las instancias contenidas dentro de los conceptos de la ontología correspondan con los valores de los parámetros del Servicio Web, para esto se establecieron dos tipos de validaciones, una de anotaciones sobre parámetros de entrada y una sobre parámetros de salida, siguiendo lo especificado en [67]. Estos dos tipos de validaciones se explican más a detalle en las sub secciones 3.2.2.6.1 y 3.2.2.6.2.

En el presente trabajo se planteó la necesidad de agregarle un estado a cada anotación según las etapas de la validación que va completando, debido a que este estado permite un análisis más detallado de los resultados obtenidos del proceso de anotación. En la tabla 3.1 se muestran los estados considerados para las anotaciones, estos estados ayudan a entender de mejor manera los resultados del proceso de anotación.

Estado	Descripción
0	Anotación inválida, tiene instancias pero no corresponden con el servicio Web.
1	Anotación no validable, el concepto no tiene instancias dentro de la ontología
2	Anotación no validable, errores en las consultas al SPARQL Endpoint (Timeout).
3	Anotación válida.
4	No anotable, Parámetro ignorado por el usuario. (Anotación especial)

Tabla 3.1: Estados definidos para las anotaciones

<sup>17</sup><http://sourceforge.net/projects/simmetrics/>

### 3.2.2.6.1. Validación de entradas

La validación de los parámetros de entrada se basa en la capacidad de invocar el servicio cambiando los valores de sus parámetros de entradas por instancias tomadas de la ontología. Si el servicio da una respuesta a la invocación y esta tiene la misma estructura base (descripción sintáctica) que la invocación inicial se considera que la invocación es correcta y por tanto las anotaciones sobre las entradas son válidas. Si por el contrario, la invocación no puede realizarse, o bien la respuesta obtenida posee una estructura diferente a la original (la respuesta del servicio tiene mensajes de error) las anotaciones son inválidas. Se plantearon cuatro fases para la validación de entradas: obtención de instancias, generación de invocaciones de validación, invocación y filtrado de invocaciones correctas.

**Obtención de instancias:** se realiza mediante consultas al SPARQL Endpoint de la ontología, estas consultas proveen una lista de instancias (tamaño definido por el usuario) con las cuales se generan las invocaciones de validación.

Las consultas que se usaron para la extracción de instancias se muestran en el segmento de código 3.1. En esta consulta el valor de *ConceptURI* corresponde a la URI del concepto del cual se extraen las instancias(Variable ?instance).

```

1 prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 select distinct ?instance
3 where {
4   ?object <ConceptURI> ?instance filter isLiteral(?instance)
5 } union {
6   ?object2 <ConceptURI> ?resource .
7   ?resource rdfs:label ?instance
8 } union {
9   ?object3 a <ConceptURI> .
10  ?object3 rdfs:label ?instance
11 } FILTER (lang(?instance) = ' ' || lang(?instance) = 'en')
12 }
```

Segmento de Código 3.1: Consulta SPARQL para extracción de instancias

Nótese que esta consulta considera tres posibilidades: primero, el concepto es una propiedad que tiene por instancias valores terminales (cadenas de texto, números, fechas); segundo, el concepto es una propiedad que tienen por instancias a objetos. En este caso se busca una propiedad dentro de dichos objetos la cual tenga valores terminales y presente al contenido del objeto.

Por ejemplo, una película (Objeto de la clase Movie) puede ser representada por su título (Propiedad: Title). Para determinar estas propiedades se utiliza el

vocabulario base de las ontologías, donde se definen propiedades para representar objetos. En la tabla 3.2 se muestra el conjunto de vocabularios base con sus propiedades 'representativas' utilizadas en la presente tesis; tercero, el concepto es una clase que tiene por instancias objetos, en este caso se aplica lo mismo que se explicó en el punto dos, es decir, se accede al valor de las instancias a través de una propiedad.

Vocabulario	Prefijo	Propiedad
RDFS	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>	rdfs:label
FOAF	<a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a>	foaf:name
Dublin Core	<a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a>	dc:title
Dublin Core	<a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a>	dct:title
SKOS	<a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a>	kos:prefLabel

Tabla 3.2: Vocabularios soportados para la extracción de instancias

**Generación de las invocaciones de validación:** Consiste en crear nuevas invocaciones al Servicio Web donde se reemplacen los valores originales de las entradas del Servicio Web por instancias obtenidas de las anotaciones sobre las entradas del Servicio, es decir, invocaciones al Servicio Web con las instancias obtenidas de la ontología.

Para la generación de las invocaciones se decidió utilizar dos estrategias: *simple* y *combinatoria*. La generación simple se basa en el reemplazo del valor de una sola entrada a la vez, dejando las entradas restantes con su valor inicial (Valor de la invocación inicial), con este método las invocaciones crecen linealmente con respecto al número de instancias obtenidas de la ontología. La generación combinatoria crea todas las combinaciones posibles entre las instancias obtenidas de la ontología y los valores iniciales de los parámetros, con este método el número de invocaciones generadas crece exponencialmente según el número de instancias obtenidas de la ontología.

Una vez generadas las invocaciones de validación se procede a ejecutarlas, para esto se llama a los componentes de invocación y descripción sintáctica. Los resultados de todas estas invocaciones son analizados y filtrados como se explica a continuación.

**Filtrado o selección de invocaciones válidas:** El filtrado de las invocaciones válidas dentro del conjunto de invocaciones realizadas se basa en dos aspectos: respuesta HTTP y el número de parámetros obtenidos en cada invocación.

Si una respuesta HTTP no tiene un código de éxito se considera a la invocación como fallida, caso contrario se continúa con el siguiente filtro, que es el

número de parámetros. Si el número de parámetros de salida obtenidos en la invocación de validación es menor al número de parámetros de salida obtenidos en la invocación inicial se considera como fallida, este último filtro se empleó porque varios Servicios Web no retornan códigos de error HTTP en las invocaciones fallidas, simplemente envían un nuevo formato (reducido) de respuesta con las causas del error.

Finalmente, si una invocación pasa los filtros antes mencionados, es considerada correcta y las anotaciones (origen de las instancias) de sus entradas son actualizadas en su estado como entradas válidas. Adicionalmente se agregan las nuevas instancias obtenidas al modelo de objetos, las mismas que servirán para la validación de parámetros de salida. Un ejemplo gráfico del proceso de validación de entradas se muestra en la figura 3.12.

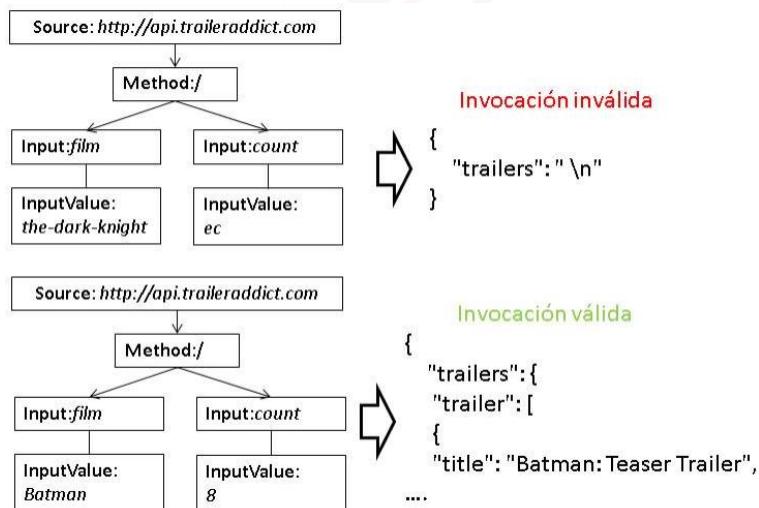


Figura 3.12: Ejemplo de ejecución del componente de validación de entradas

### 3.2.2.6.2. Validación de salidas

La validación de las anotaciones sobre los parámetros de salida del servicio consiste en verificar si las instancias de dichos parámetros (obtenidas de las invocaciones) corresponden con las instancias de las ontologías de los conceptos con los que se anotaron. Esta verificación se puede lograr con la búsqueda de las instancias del Servicio Web dentro de las instancias de los conceptos de la ontología, a través de su SPARQL Endpoint [67].

Se planteó que para la ejecución de esta validación se debe verificar cada anotación realizada sobre los parámetros de salida, es decir, buscar todas las instancias

de los parámetros de salida (de todas las invocaciones) sobre todos los conceptos con los cuales se anotó dicho parámetro. Adicionalmente, se estableció que si un determinado número de instancias (del parámetro) puede ser encontrado dentro de las instancias del concepto anotado se considera a la anotación como válida. En la figura 3.13 se muestra un ejemplo del proceso de validación de salidas.

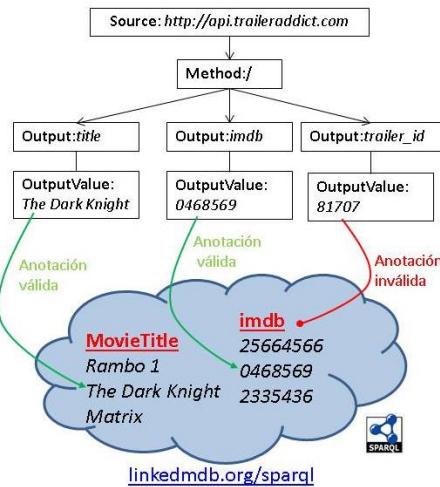


Figura 3.13: Ejemplo de ejecución del componente de validación de salidas

Para determinar si una instancia (valor) está dentro de un concepto se debe utilizar el SPARQL Endpoint de la ontología, a través de consultas. Dichas consultas deberían definir si un valor es o no instancia de un concepto determinado. Con este fin de creó la consulta del segmento de código 3.2, la cual retorna *true* en caso que el valor corresponda al concepto y *false* si no.

```

1 prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 ask {
3 { ?object <ConceptURI> ?instance filter isLiteral(?instance)
4 } union {
5 ?object2 <ConceptURI> ?resource .
6 ?resource rdfs:label ?instance
7 } union {
8 ?object3 a <ConceptURI> .
9 ?object3 rdfs:label ?instance
10 }
11 FILTER ( regex(str(?instance), 'Instance Value', 'i') || regex('
12 Instance Value', str(?instance), 'i') )
}
  
```

Segmento de Código 3.2: Consulta para validación de instancias

En la consulta de verificación de conceptos al igual que en la consulta de extracción de instancias se han considerado tres posibles escenarios: propiedades con valores terminales, propiedades con valores complejos (objetos) y clases. Además, se utiliza la función *regex* que permite realizar búsquedas sobre texto más flexibles, ignorando mayúsculas y minúsculas, realizando comparaciones tipo *contains*, todo esto con el objetivo de mejorar el número de anotaciones validadas.

En cuanto al número de instancias necesarias para determinar cuándo se considera una anotación válida se decidió utilizar dos criterios. El primero consiste directamente en establecer un número mínimo de instancias verificadas que debe tener cada anotación. El segundo, es establecer un porcentaje de instancias verificadas con respecto al total de instancias de obtenidas para cada parámetro. El usuario puede elegir usar uno de estos dos criterios a través de la interfaz gráfica, según su elección.

El resultado final de este componente de validación es el modelo de objetos completo incluyendo cambios en el estado de las anotaciones tanto sobre parámetros de entrada como de salida. Este modelo se almacena dentro de un repositorio para su posterior utilización.

### Consideraciones de implementación

- Para la ejecución de la validación de entradas se estableció un tiempo de espera entre invocaciones de validación. Este tiempo es definido por el usuario desde la interfaz y evita que el servidor del Servicio Web considere como un ataque de denegación de servicios las peticiones.
- Se creó una caché de consultas de instancias válidas (Validación de salidas) dentro de conceptos con los siguientes campos *Cache* (*Value*, *IsIn*, *Concept*). Esto con el fin de evitar repetir consultas en los SPARQL Endpoints que tienen tiempos de respuesta considerables.

#### 3.2.2.7. Componente de persistencia

El componente de persistencia se encarga de almacenar el modelo de objetos resultante en una base de datos relacional. Dicho modelo contiene toda la información generada por el proceso de anotación y por tanto debe ser almacenado. Para ello se mapearon las clases del modelo de objetos como tablas de una base de datos. Dicho mapeo fue directo, es decir cada clase fue mapeada a una tabla con su mismo nombre y atributos.

## Consideraciones de implementación

- Para simplificar las actividades de persistencia de utilizó la librería ORM-Lite<sup>18</sup>. Por ser fácilmente integrable dentro del Bus de Servicios.

### 3.2.3. Desarrollo de la interfaz gráfica para la plataforma de anotación

Para facilitar la interacción con la plataforma como se menciona en 3.1.2, se creó de una interfaz gráfica, la cual permite el acceso a todas las funcionalidades que se han desarrollado. En esta sección, se tratará el desarrollo de dicha interfaz considerando para ello dos aspectos principales: análisis de funcionalidades requeridas y las consideraciones de implementación.

#### 3.2.3.1. Análisis de funcionalidades

Dentro de la implementación del proceso de anotación semántica se reconocieron varias necesidades de interacción del usuario con el Bus de Servicios así como con el repositorio de Servicios anotados.

#### Iniciar el proceso de anotación semántica de un Servicio Web

Esta funcionalidad permite a los usuarios enviar un Servicio Web al Bus de Servicios para la ejecución del proceso de anotación, para lo cual es requerido el ingreso de un conjunto de parámetros por parte del usuario (detallados en la tabla 3.3). La interfaz gráfica desarrollada para esta funcionalidad se muestra en la figura 3.14.

**Semantic RESTful Web Services Annotation**

<b>Menu</b> SWSA Add Web Service ESB Status Web Service Repository SPARQL Endpoints Search	WS URL: <input type="text" value="http://api.eventful.com/json/events/search?app_key=JQVnXsZrR2Gsvg3X&amp;keywords=books&amp;loc"/> Description: <input type="text" value="Eventful Web Service"/> Ontology Selection Method: <input type="button" value="Solr 1"/> SPARQL Endpoint: <input type="text"/> Ignore parameters: <input type="text"/> Match Threshold(Inputs): <input type="text" value="0.70"/> Match Threshold(Outputs): <input type="text" value="0.70"/> Validation Instances: <input type="text" value="10"/> Invocations generation: <input type="button" value="Simple"/> Query Timeout: <input type="text" value="300"/> Invocation delay (ms): <input type="text" value="5000"/> Output Validation Threshold (Num, Percentage): <input type="text" value="1"/> <input type="button" value="Start Annotation"/>
---	--

Figura 3.14: Captura de la funcionalidad para iniciar el proceso de anotación

<sup>18</sup><http://ormlite.com/>

Parámetro	Descripción	Ejemplo
URL de invocación del Servicio	URL del Servicio, la cual posea parámetros de entrada del Servicio con instancias (valores iniciales).	<code>http://api.eventful.com/json/events/search?app\_key=JQVnXsZrR2Gsvg3X\&amp;keywords=books\&amp;location=San+Diego\&amp;date=Future</code>
Nombre/Descripción	Nombre o descripción corta del Servicio que permita diferenciarlo de los demás Servicios del repositorio.	Eventful Web Service
Método de selección de ontologías	Identificador del método de selección automática de ontologías a ser usado para el proceso de anotación.	Solr 1
Ontología contra la cual anotar	Es opcional, solo se especifica si el usuario quiere seleccionar manualmente el SPARQL Endpoint de la ontología a ser usada en la anotación.	<code>http://dbpedia.org/sparql</code>
Parámetros ignorados	Lista de parámetros del Servicio Web que el usuario desea ignorar para el proceso de anotación	Id,search_time,ver,out
Umbral de Matching para entradas.	Valor a ser usado como umbral dentro del componente de matching con los parámetros de entrada.	0.80
Umbral de Matching para salidas.	Valor a ser usado como umbral dentro del componente de matching con los parámetros de salida.	0.80
Número de instancias de validación	Número de instancias de los conceptos a ser usadas en el componente de validación de entradas.	3
Estrategia de generación de invocaciones de validación	Forma en la que las instancias de los conceptos se utilizarán en el componente de validación de entradas, para la generación de invocaciones de validación.	Simple
Tiempo de espera para las consultas a la ontología	Tiempo máximo de espera para la ejecución de consultas al SPARQL Endpoint de la ontología en segundos	60
Espera entre invocaciones	Tiempo de espera entre invocaciones al Servicio Web, usado en el componente de invocación, definido en milisegundos.	500
Umbral de validación de salidas	Número o porcentaje de instancias de los parámetros del Servicio Web necesarias para validar las anotaciones en el componente de validación de salidas	2 %

Tabla 3.3: Lista de configuraciones para el proceso de anotación

### ***Log* del proceso de anotación**

A través de esta funcionalidad el usuario es capaz de visualizar el estado de los procesos de anotación iniciados previamente y que se encuentran en ejecución dentro del Bus de Servicios. La información se visualiza en cuadros de texto que muestra los eventos y acciones realizadas dentro del Bus de Servicios. El uso de *logs* aparte de ser informativo es utilizado para la depuración de errores, por lo

que se creó una sección independiente que captura exclusivamente los problemas encontrados o excepciones (*Error Log*). En la figura 3.15 se muestra la interfaz gráfica desarrollada para manejo de *logs* del proceso de anotación.



Figura 3.15: Captura de la funcionalidad de monitoreo del proceso de anotación

### Visualización de los Servicios dentro del Repositorio

Esta funcionalidad permite la visualización de los Servicios Web que han terminado el proceso de anotación (Servicios anotados) dentro de una tabla resumen, donde se muestran los parámetros de dichos servicios, anotaciones resultantes del proceso de anotación e instancias de ejemplo.

Los columnas consideradas para la tabla de visualización de Servicios Web anotados se muestran en la tabla 3.4. Adicionalmente, a la tabla resumen se le agregaron controles gráficos desplegables pensados para que la navegación por parte del usuario sea más fácil. En la figura 3.16 se muestra una captura del funcionamiento de esta pantalla.

Columna	Descripción	Ejemplo
Parámetros	Muestra el nombre de los parámetros del Servicio Web	City
Concepto	Muestra la URI del concepto con el cual se anotaron los parámetros del servicio	<a href="http://dbpedia.org/ontology/location">http://dbpedia.org/ontology/location</a>
Tipo de anotación	Indica el tipo de anotación que se alcanzó.	Validada
Ontología	Es la URL del SPARQL Endpoint de la ontología con la que se anotó el servicio.	<a href="http://dbpedia.org/sparql">http://dbpedia.org/sparql</a>
Valores del Concepto	Instancias del concepto con el que se anotaron los parámetros, tomadas desde la Ontología.	Cuenca-Ecuador
Valores del Servicio	Instancias o valores de los parámetros del Servicio Web.	Cuenca

Tabla 3.4: Información utilizada para la visualización de Servicios Anotados

**Semantic RESTful Web Services Annotation**



Web Service					
Parametro	Concepto	Tipo	Endpoint	Valores Concepto	Valores Servicio
Entradas	-	-	-	-	-
plot	-	-	-	-	full,full,full,full,
r	-	-	-	-	json,json,json,
^t	-	-	-	-	Batman,Fearless,Godzilla,...
Anotacion 1	http://purl.org/dc/terms/title	Validada	http://data.linkedmdb.org...	buffy the vampire slayer,...	-
Anotacion 2	http://www.w3.org/1999/02/22-rdf-sy...	Sin Instancias	http://data.linkedmdb.org...	-	-
y	-	-	-	-	2015,2015,2015,2015,
Salidas	-	-	-	-	-

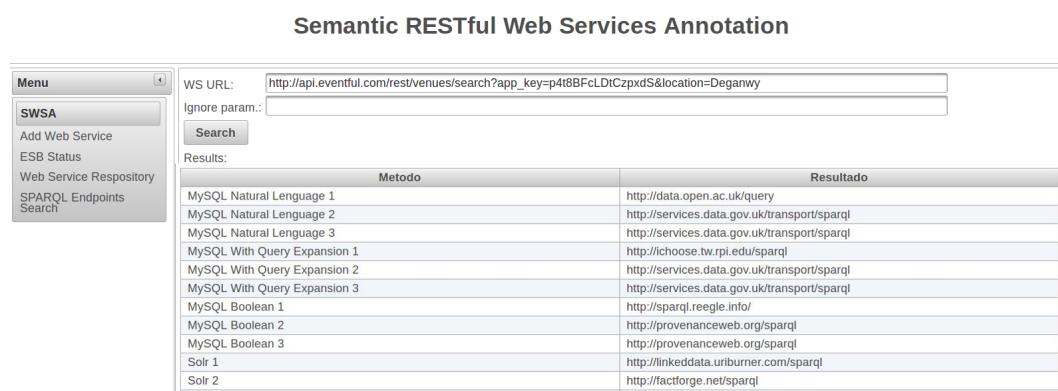
Figura 3.16: Captura de la funcionalidad de visualización de Servicios Anotados

### Búsqueda de ontologías

Esta funcionalidad permite la ejecución de los métodos de búsqueda de ontologías directamente, es decir, buscar ontologías adecuadas para un Servicio Web sin necesidad de ejecutar todo el proceso de anotación.

Para iniciar la ejecución de las búsquedas de ontologías se consideraron dos elementos de los Servicios Web que son: La URL de invocación a los servicios y una lista de parámetros del servicio a ser ignorados para la ejecución de la búsqueda (opcional). Estos dos elementos son enviados al Bus de Servicios que ejecuta la búsqueda utilizando el componente de selección de ontologías. Los resultados de la búsqueda son retornados a la interfaz donde se presentan en una tabla en la que se disponen todos los métodos de búsqueda desarrollados con las ontologías que fueron seleccionados. En la figura 3.17 se muestra un ejemplo de la ejecución de esta funcionalidad.

**Semantic RESTful Web Services Annotation**



Metodo	Resultado
MySQL Natural Language 1	http://data.open.ac.uk/query
MySQL Natural Language 2	http://services.data.gov.uk/transport/sparql
MySQL Natural Language 3	http://services.data.gov.uk/transport/sparql
MySQL With Query Expansion 1	http://choose.tw.rpi.edu/sparql
MySQL With Query Expansion 2	http://services.data.gov.uk/transport/sparql
MySQL With Query Expansion 3	http://services.data.gov.uk/transport/sparql
MySQL Boolean 1	http://sparql.reegle.info/
MySQL Boolean 2	http://provenanceweb.org/sparql
MySQL Boolean 3	http://provenanceweb.org/sparql
Solr 1	http://linkeddata.uriburner.com/sparql
Solr 2	http://factforge.net/sparql

Figura 3.17: Captura de la funcionalidad de búsqueda de ontologías

### 3.2.3.2. Consideraciones de implementación

- Para la implementación de esta aplicación Web se utilizó JSF con Prime Faces.
- Para la extracción de información del repositorio (Base de datos) se utilizó JPA, debido a que esta librería facilita la conexión con las bases de datos usando ORM.
- Con el fin de facilitar la navegación entre los logs producidos por los distintos procesos de anotación que se ejecutan en el Bus de Servicios se decidió asignar un número de proceso a cada instancia.
- Para la conexión entre el Bus de Servicios con la aplicación Web desarrollada se creó un Servicio Web que permita el intercambio de información mediante peticiones HTTP. Este servicio es utilizado tanto para el envío de nuevas tareas de anotación (anotación de servicios y búsqueda de ontologías) al Bus de Servicios así como la recepción de resultados del Bus de Servicios (logs y resultados de las búsquedas).

## Capítulo 4

---

### Selección automática de ontologías

---

Un elemento esencial dentro del proceso de anotación semántica que no ha sido considerado en [21] y que forma parte de los aportes destacados en la presente tesis, es la etapa de búsqueda y selección automática de ontologías. Con el desarrollo de esta etapa se busca brindar un mayor nivel de automatización al proceso de anotación semántica de los Servicios Web, pues tiene como objetivo encontrar la ontología que mejor represente al servicio con el cual realizar el proceso anotación.

La implementación de dicha etapa se llevó a cabo mediante el desarrollo de un componente que permita la obtención de esquemas ontológicos de forma automática en base a las características de un servicio. Para la lógica de dicho componente, se requirió el planteamiento de varios métodos que sean capaces de automatizar la obtención de ontologías. Esto debido a que la selección automática de ontologías aún está en proceso de investigación y no existen métodos de selección de ontologías disponibles. Se plantearon un conjunto de métodos candidatos, a los cuales se analizó para determinar su viabilidad para resolver el problema, fueron implementados, evaluados y jerarquizados con el fin de determinar el método que ofrece los mejores resultados (ver capítulo 5).

Teniendo en cuenta los objetivos mencionados en los cuales un método debe obtener las ontologías que mejor representen un servicio en cualquier dominio y que los métodos seleccionados deben integrarse con el resto de componentes en el proceso de anotación, se han logrado detectar un conjunto de requerimientos y consideraciones base, que deben cumplir los métodos para hacer posible su utilización, los cuales se describen a continuación.

- Los repositorio de donde se extraiga las ontologías deben ser de diversos ámbitos (deportes, ciencia, geográficos, etc.).

- Las ontologías deben seleccionarse en base al mejor nivel de *matching* entre la descripción sintáctica del Servicio Web a anotar y los conceptos de la ontología (Esquema).
- La ontología seleccionada debe contener instancias, pues la etapa de validación de las anotaciones semánticas requiere de estas instancias para la verificación de las anotaciones.

## 4.1. Planteamiento de métodos de búsqueda de ontologías

A continuación se describe un conjunto de métodos planteados como candidatos para formar parte del componente de selección automática de ontologías. Para cada uno de los métodos se analiza las ventajas y sus posibles complicaciones para así determinar cuáles pueden ser implementados y de los seleccionados cuáles son los que aportan efectivamente con los mejores resultado en el proceso de anotación semántica de Servicios Web.

Para el desarrollo de los métodos se consideró dos elementos de entrada principales, obtenidos de etapas anteriores como son: los atributos de entrada del servicio y los atributos de salida, enriquecidos respectivamente con sinónimos y sugerencias. Dichos datos serán la base de trabajo para todos los métodos y con los que se debería encontrar la ontología que mejor represente al servicio.

## 4.2. Búsqueda de ontologías en repositorios externos

Cuando se requiere realizar búsquedas de ontologías comúnmente se recurre a repositorios de ontologías que son accesibles en la Web y que contienen datos abiertos (OPEN DATA). Los datos abiertos se refieren a información liberada por los gobiernos y otras instituciones que ponen a disposición del público en general datos que son recopilados y que pueden ser de gran ayuda en proyectos de investigación. Dentro de los repositorios de ontologías a esto tipo de datos se les puede encontrar bajo la denominación de *Linked Open Data* (LOD). LOD<sup>1</sup> es un proyecto en el que se han recopilado datos libres, que siguen la recomendaciones

---

<sup>1</sup><http://linkeddata.org/>

de Linked Data y que además puedan ser accesibles al público en formato estándar para intercambio de información como RDF, OWL, etc .

El acceso a los datos de LOD se realiza principalmente por medio SPARQL Endpoints, sin embargo debido a la falta de una interfaz que permita la navegación intuitiva en la información, se consideró hacerlo por un acceso externo. Este acceso externo se realizó mediante el repositorio y catálogo Datahub<sup>2</sup>, siguiendo la recomendación de la wiki del mismo proyecto. Datahub es un repositorio abierto y poderoso basado en la plataforma C-KAN<sup>3</sup>, que permite entre algunas de las funciones destacadas la administración de ontologías y *datasets*, búsqueda avanzada de datos, asignación de metadatos, obtención estadísticas y gráficos, entre muchas otras. Características que facilitan en gran medida la búsqueda y administración de la información dentro de este repositorio. Uno de las ventajas adicionales que presenta y que beneficia al uso de la plataforma, es la de capacidad de acceder a *datasets* que no se encuentran dentro de LOD pero que aun así son de libre acceso, ventaja que aporta a la variedad de datos disponibles para la búsqueda de ontologías.

Muchos de los métodos de búsqueda y selección de ontologías que se describen a continuación, utilizan Datahub como base para la obtención de resultados. Mediante la combinación de los métodos de búsqueda conjuntamente con la funcionalidades que ofrece el buscador de Datahub se pretende explotar el potencial de la herramienta para verificar su factibilidad dentro del proceso de búsqueda automática de ontologías.

#### 4.2.1. Elementos para la organización de datos en el repositorio

Como se mencionó en la sección 4.2, Datahub es un catálogo de información de ontologías, por lo tanto requiere de un conjunto de atributos y metadatos para poder almacenar y organizar la información. Dentro de los elementos más importantes que se agregan con cada *dataset*<sup>4</sup> para facilitar su identificación se encuentran etiquetas como :

**Organization** : Describe la información de organizaciones que son las responsables de los *datasets*, en lugar de una sola persona, por ejemplo Open Data Ecuador, Linked Open Data Cloud, etc.

---

<sup>2</sup><http://datahub.io/>

<sup>3</sup><http://ckan.org/>

<sup>4</sup>Dataset: Colección de datos

**Groups** : Describen a un conjunto de *datasets* agrupándolos bajo una denominación, como ejemplo pueden llevar la descripción de un proyecto específico al que pertenecen.

**Tags** : Son un conjunto de palabras descriptivas acerca del conjunto de datos, similar a las keywords dentro de las publicaciones.

**Formats** : Indican el formato en el que están dispuestos los datos, pueden ser RDF, CSV, OWL, SPARQL ENDPOINT, entre otros.

**Description** : Contiene una breve descripción del contenido del *dataset*.

**Title** : Nombre del *dataset*

#### 4.2.2. Búsqueda directa utilizando Datahub

La búsqueda de ontologías por medio de Datahub se puede realizar de dos maneras mediante un cuadro de búsqueda en la página principal o mediante un API Endpoint<sup>5</sup>. En el presente trabajo debido a que se requiere añadir el proceso de búsquedas de ontologías dentro del proceso de anotación semántica de servicios de forma automática, se optó por utilizar el acceso a los *datasets* mediante el API. El formato que se utiliza para las llamadas al servicio de búsqueda es el siguiente:

```
http://datahub.io/api/search/dataset?q=[keywords]&f1=name,  
res_url,res_format
```

Donde los parámetros del API que se puedan ingresar a las búsquedas, son básicamente las palabras clave “*keywords*” que se requieren buscar dentro del repositorio, así como los campos que se desean mostrar en la respuesta. A los criterios de búsqueda mencionados se ha planteado también incluir la palabra “sparql” dentro de las palabras clave, pues con la incorporación de dicha palabra se puede solicitar al repositorio que incluya en la búsqueda *datasets* que contengan enlaces SPARQL Endpoint.

#### Pruebas mediante búsqueda directa

Como prueba inicial de búsqueda directa utilizando el API Endpoint de Datahub, se realizó una consulta básica tomando como datos de entrada las palabras

---

<sup>5</sup><http://docs.ckan.org/en/latest/api/legacy-api.html>

de contenidas en la tabla 4.1. Dichas palabras están conformadas por los parámetros del servicio Eventful obtenidos mediante la llamada descrita en el segmento de código 4.1 , luego de haber pasado los procesos de descripción sintáctica y enriquecimiento tratados en las secciones 3.2.2.2 y 3.2.2.3. Para dicha consulta como se puede ver en el segmento de código 4.2, no se han encontrado resultados por parte del buscador, por lo que se puede deducir que un número grande de palabras como los utilizados en dicha consulta puede perjudicar a la cantidad de datos obtenidos como respuesta.

```
http://api.eventful.com/rest/venues/search?app_key=p4t8BFcLDtCzpxdS&location=Madrid
```

Segmento de Código 4.1: Consulta ejemplo sobre el servicio Eventful

```
app_key location placement locating position positioning page_size page_size city_name region_abbr region_abbr venue_name venue_name total_items total_items items page_number @version last_item search_time search_time country_name country_name name page_count region_name timezone height tallness acme elevation peak created description verbal_description form kind sort name gens figure public figure epithet first_item first_item page_items longitude angular_distance event_count comment_count trackback_count trackback_count count width breadth dimension image mental image persona picture icon caption subtitle legend exception interlingual renditon country_abbr2 url URL uniform_resource_locator universal_resource_locator address venue_type venue_type country_abbr country_abbr address computer_address speech destination name_and_address link_count postal_code postal_code owner_proprietor possessor business_man individual geocode_type geocode_type @id latitude line_of_latitude parallel_of_latitude parallel_ambit
```

Tabla 4.1: Palabras entrantes al componente del Servicio Eventful

```
{"count": 0, "results": []}
```

Segmento de Código 4.2: Respuesta en JSON por búsqueda directa

Frente al resultado poco favorable al ingresar un gran número de palabras en la búsqueda, se presenta una nueva prueba, en el que se pretende observar el desempeño del buscador frente a un número reducido de palabras. En este caso el número de palabras que se utilizó son las palabras más relevantes según el criterio de un experto considerando que el servicio trata sobre eventos en distintas

localidades del mundo. Esta variación se realizó para comprobar cómo afecta la cantidad de las palabras ingresadas en el resultado devuelto por el buscador. Los resultados de las pruebas se pueden ver en la tabla 4.2, hay que tener en cuenta que los resultados obtenidos de la consulta son *datasets*, los cuales tienen enlaces a diversos recursos tanto SPARQL, como de otro tipo (RDF, Excel, CSV) detallados en la última columna de la tabla.

Palabras	Num resultados	Nombre datasets (5 primeros)	Num enlaces a SPARQL Endpoints	Num enlaces a otros recursos
location position event address time-zone placement sparql	1	http://www.ijape.org/	0	74
location event address sparql	13	Inertia-energy-consumption-example-linked-data semantic-web-dog-food symbol-signs vivo agrovoc-skos	1 1 0 0 1	4 2 1 6 32
location event sparql	92	Semantic-profiling-network budapest-terfigyelo-koltsegek-2007-08 bioportal-teo bioportal-iev bioportal-aero	1 0 1 1 1	0 1 2 2 2

Tabla 4.2: Pruebas mediante Búsqueda directa en Datahub (número reducido de palabras)

### Conclusiones para la búsqueda directa

Usar el motor de búsqueda de Datahub permite la reutilización de un servicio ya desarrollado, lo que representa un ahorro considerable de tiempo de implementación, a diferencia de generar uno propio. Además mediante el buscador del repositorio, se puede obtener los enlaces hacia los recursos como RDF, OWL, ejemplos, API SPARQL Endpoint permitiendo acceder directamente a dichos recursos dentro de los *datasets*. Sin embargo, existen algunos problemas que tiene el método, como el número reducido de *datasets* que presenta como resultados conforme se aumentan el número de palabras en la búsqueda, lo que es hasta cierto punto justificable debido a que el buscador está diseñado para mostrar solo los resultados que cumplan con todos los criterios de entrada.

#### 4.2.3. Búsquedas por Datahub mediante la adaptación de entradas

El problema presentado en la búsqueda directa mediante Datahub debido al número grande de entradas, ha generado la necesidad de plantear algunas técnicas que tratan de mejorar la calidad de datos que ingresan a la búsqueda con el fin de obtener mejores resultados. Estas técnicas principalmente están basadas en el uso de herramientas de procesamiento de lenguaje natural, con el cual se pretende simular a un experto humano, que seleccionará los datos más relevantes antes de ingresarlos al buscador, similar al proceso realizado sobre la tabla 4.2 pero sin intervención manual. Para realizar dicha función se hará uso de dos herramientas conocidas en este ámbito como AlchemyApi<sup>6</sup> y Gate<sup>7</sup>.

##### AlchemyApi

AlchemyApi es una tecnología de procesamiento de lenguaje natural y una máquina de aprendizaje de algoritmos. Es utilizada principalmente para la extracción semántica de metadatos de contenidos como: información de personas, lugares, tópicos, hechos, relaciones, autores y lenguajes. Existe una versión demo y una versión profesional de la herramienta, pero debido a que sólo se necesita para la realización de pruebas, la función demo presta todas las funcionalidades requeridas.

##### GATE

*General Architecture for Text Mining* (GATE) es una plataforma abierta para el procesamiento de lenguaje natural y extracción de información (*Information Extraction - IE*). Provee una serie de herramientas que facilitan actividades de extracción de conocimiento, anotación semántica de contenidos, indexación de textos, etc. De estas las más destacadas entre la comunidad científica han sido las herramientas de extracción de información, por lo que se han agrupado en un sub proyecto llamado ANNIE (*A Nearly-New Information Extraction System*) [16]. ANNIE provee funciones de reconocimiento de nombre de entidades (*Named Entity Recognition - NER*) y categorización de dichas entidades bajo la clasificación de la *Message Understanding Conference* (MUC) [57]. GATE al tratarse de un proyecto de software libre puede ser incorporado a otros proyectos fácilmente

---

<sup>6</sup><http://www.alchemyapi.com/>

<sup>7</sup><https://gate.ac.uk/>

usando la distribución GATE Embedded, que consiste en un conjunto de librerías JAR<sup>8</sup>, las cuales pueden ser usadas desde los lenguajes de programación soportados por la Java Virtual Machine <sup>9</sup>.

#### 4.2.3.1. Búsqueda por dominio

Una de las alternativas para la búsqueda directa dentro de los *datasets*, que permite superar el problema de tener muchos parámetros dentro de la búsqueda, es caracterizarlos primero dentro de un grupo o clasificación. Se puede aplicar esta idea tomándola del propio Linked Open Data, que como parte de su organización agrupa los *datasets* que contiene por un dominio específico. Esto permite a LOD obtener estadísticas e información del tipo de datos que manejan y conocer comó avanza este.

Las clasificaciones por dominio que se pueden encontrar desde la página LOD-Cloud<sup>10</sup> son :

- Media
- Geografía
- Gobierno
- Publicaciones
- Dominio cruzado
- Ciencias de la vida
- Contenido generado por el usuario.

La propuesta de búsqueda mediante dominio consiste en utilizar herramientas que caractericen los atributos de entrada y salida que ingresan al componente de selección de ontologías para obtener una clasificación que las englobe. Dicha clasificación puede ayudar a localizar con el buscador de Datahub, el *dataset* que tengan una relación mayor con dicha caracterización. Con el método de búsqueda por dominio se reduce la cantidad de palabras necesarias dentro del buscador, con lo que se pretende aumentar la cantidad de datos encontrados, y estos a su vez estarán relacionados con el tema de mayor peso dentro del conjunto de datos.

---

<sup>8</sup>Java ARchive - Formato de paquetes soportado por la plataforma Java

<sup>9</sup><https://www.java.com/es/>

<sup>10</sup><http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

Para realizar la clasificación se utiliza la herramienta AlchemyApi debido a que posee una taxonomía con más de 1000 categorías en el idioma inglés además de ser capaz de asignar sub-categorías hasta dos niveles inferiores frente a la categoría principal. En este caso GATE no se ha considerado debido a que posee una clasificación distinta por entidades que se presenta en la tabla 4.7

### Pruebas de búsqueda por dominio

A continuación se presenta una prueba realizada utilizando la demo de Alchemy, con el conjunto de datos de la table 4.1 como entrada. Las palabras utilizadas pertenecen al conjunto de palabras extraídas del servicio Eventful utilizado en las pruebas anteriores.

Clasificación	Puntaje
/business and industrial	0.124375
/technology and computing/software/databases	0.097077
/art and entertainment/visual art and design/design	0.0921094

Tabla 4.3: Taxonomías resultante utilizando AlchemyApi

Los resultados obtenidos en la tabla 4.3 muestran 3 clasificaciones con su respectivo puntaje. En primer lugar ubica a Negocios e industria, en segundo a tecnologías y computación, y en tercero arte y entrenamiento. Todos las clasificaciones presentan un puntaje muy bajo respecto a la máxima de 1, demostrando que la clasificación no es confiable. Se puede comprobar este hecho dado que ninguna clasificación se aproxima a describir el servicio Eventful, el cual trata sobre organización de eventos de distintas partes del mundo. La categoría más cercana dentro de AlchemyApi que se relacionaba con el servicio y que se hubiera esperado obtener es *Art and entertainment/ show and events*.

### Conclusiones de la búsqueda por dominio

Alchemy es una herramienta desarrollada para describir las características de un texto acerca de un tópico específico como un artículo o una noticia, pero no está adaptada para soportar la clasificación para un conjunto de palabras sin estructura, por lo que los resultados no llegan a ser muy exactos para el caso de esta propuesta. Es necesario reconocer además que no existe una misma clasificación entre las taxonomías empleadas en Alchemy respecto a la clasificación por dominios de LOD por lo que no se puede esperar que los resultados vayan a coincidir en gran medida.

#### 4.2.3.2. Búsqueda por palabras clave

El conjunto de parámetros que sirven como base para la búsqueda de ontologías está conformado por un gran número de palabras, debido a que son el resultado de los atributos de entrada y salida, sinónimos y sugerencias de un servicio. Muchas de estas palabras hablan sobre conceptos similares y presentan una estrecha relación entre sí, mientras que en otros casos también existen palabras que no guardan mucha relación con el resto de parámetros complicando el proceso de búsqueda. En el presente planteamiento se propone disminuir la cantidad de datos a ingresar en la búsqueda dentro de Datahub mediante la obtención solamente de aquellas palabras más relevantes dentro del conjunto de datos y que mediante sus relaciones puedan describir a las demás. Para realizar esta tarea se usará la función *keywords* de Alchemy, que ordena las palabras por su importancia a base de reconocimiento de lenguaje natural y medios estadísticos.

#### Pruebas de búsqueda por palabras clave

Para realizar las pruebas que demuestren la viabilidad de esta propuesta, se usó una vez más las palabras descritas en la tabla 4.1, provenientes del servicio Eventful.

Keyword	Relevancia
uniform resource locator	0.910864
universal resource locator	0.905543
region abbr	0.862563
country abbr	0.818374
interlingual rendition	0.794776
trackback count	0.779166
angular distance	0.757582
verbal description	0.750786
geocode type	0.750733
page size	0.750157
public figure	0.749127

Tabla 4.4: Palabras organizadas por relevancia según AlchemyApi (10 primeras)

Las 10 primeras palabras resultantes obtenidas de la prueba mediante la función *keywords* se puede ver en la tabla 4.4. Aunque las palabras tienen una alto nivel de relevancia (mayor al 0.75) frente a las demás, estas no coinciden con las palabras seleccionadas como más relevantes por un experto humano presentadas en la tabla 4.2. Por lo tanto se puede concluir que la función de AlchemyApi aún no puede inferir de manera similar a un experto humano para este caso.

## Conclusión de la búsqueda por palabras clave

La idea central de la utilización de la función keywords de Alchemy API es la de permitir seleccionar las palabras más relevantes dentro del conjunto de datos, eliminando las palabras que no aportan mucho contenido a la búsqueda en base a un criterio. Sin embargo, la tarea de decidir qué palabra es más relevante con respecto a la otra es una tarea que depende del ámbito en el que se requiere usar, cosa que como se apreció en la sección 4.2.3.1 no puede obtener correctamente Alchemy a través de las taxonomías. Aun así, los resultados son mejores que la búsqueda por dominios, porque las palabras al menos si están relacionadas con las características del servicio. Como comprobación adicional, la búsqueda realizada por medio de Datahub con los dos primeras palabras obtenidas dieron como resultados 28 *datasets*, donde 2 de los 3 primeros contienen la etiqueta de “geográfico”, que es una de las características del servicio como se muestra en la tabla 4.5.

Recurso	Etiquetas
OECD International Development Statistics (IDS) on aid and other resource flows	access-www, aid, flows, development, size medium, todo-splitup ,transparency
Linked Crowdsourced Data	Cdc,context data cloud, cuisines, dishes, food, food and beverages, format-geo, format-gr, format-rdf, format-rdfs, <b>geographic</b> , ingredients, licence-metadata, lod, no-deref-vocab, published-by-producer, semantic services, sparql
OpenMobileNetwork	base-station, cell, format-geo, format-owl, format-rdf, <b>geographic</b> , license-metadata, lod, mobile-network, mobile-phone, no-deref-vocab, provenance-metadata, published-by-producer, sparql, wireless-network

Tabla 4.5: Recursos resultantes de la consulta a Datahub (3 primeros)

### 4.2.3.3. Búsqueda por conceptos

Cuando se requiere encontrar información de utilidad dentro de un texto grande, es común que las personas intenten reconocer la temática de la que trata y su relevancia, mediante la búsqueda de los conceptos más importantes dentro del contenido. En el siguiente planteamiento siguiendo esa idea, se propone obtener los conceptos más relevantes que posee la entrada de datos, para ser usados dentro del proceso de búsqueda en Datahub. De esta manera se trata de reducir el número de elementos dentro de la búsqueda que no aportan mucha información, esto a través de conceptos de mayor importancia. Para realizar esta tarea, se usará la función que AlchemyApi ha desarrollado para el reconocimiento de conceptos, que funciona de manera similar a como lo haría una persona, organizando los conceptos por relevancia y que brinda una característica adicional como el enlace

a recursos de Linked Data. El enlace de los conceptos que realiza AlchemyAPI los hace con los siguientes recursos :

**DBpedia**<sup>11</sup> : Un esfuerzo de la comunidad de multitud de fuentes para extraer información estructurada de Wikipedia y poner esta información disponible en la Web.

**Freebase**<sup>12</sup> : Una base de datos refinada por la comunidad de personas conocidas, lugares y cosas.

**Censo de EE.UU**<sup>13</sup> : acceso rápido y fácil a los datos sobre las personas, las empresas y la geografía.

**GeoNames**<sup>14</sup> : La base de datos geográfica GeoNames cubre todos los países y contiene más de ocho millones de nombres de lugares que están disponibles para su descarga gratuita.

**Umbel**<sup>15</sup> : Está diseñado para ayudar a interoperar contenido en la Web.

**OpenCyc**<sup>16</sup> : La puerta de entrada a todo el poder de Cyc, base de conocimientos generales más grande y más completa del mundo y motor de razonamiento de sentido común.

**YAGO**<sup>17</sup> : Una enorme base de conocimiento semántico que contiene más de 120 millones de datos sobre más de 10 millones de entidades.

**MusicBrainz**<sup>18</sup> : Una enciclopedia de música abierta que recoge metadatos de música y lo hace disponible para el público.

**CIA Factbook**<sup>19</sup> : Ofrece información sobre la historia, la gente, gobierno, economía, geografía, comunicaciones, transporte, militar, y las cuestiones transnacionales de 267 entidades del mundo.

**CrunchBase**<sup>20</sup> : La base de datos gratuita de las empresas de tecnología, personas e inversores que cualquiera puede editar.

### Pruebas de búsqueda por conceptos

Para realizar las pruebas una vez más se usó el conjunto de palabras de la tabla 4.1. Los resultados se muestran en la tabla 4.6

Concepto	Linked Data	URI	Relevancia
Uniform Resource Locator	dbpedia	<a href="http://dbpedia.org/resource/Uniform\_Resource\_Locator">http://dbpedia.org/resource/Uniform\_Resource\_Locator</a>	0.97799
	yago	<a href="http://yago-knowledge.org/resource/Uniform\_Resource\_Locator">http://yago-knowledge.org/resource/Uniform\_Resource\_Locator</a>	
Latitude	dbpedia freebase	<a href="http://dbpedia.org/page/Latitude">http://dbpedia.org/page/Latitude</a> <a href="http://rdf.freebase.com/ns/m.04gmf">rdf.freebase.com/ns/m.04gmf</a>	0.810546
Uniform Resource Identifier	dbpedia	<a href="http://dbpedia.org/page/Uniform\_resource\_identifier">http://dbpedia.org/page/Uniform\_resource\_identifier</a>	0.781394
	yago	<a href="http://yago-knowledge.org/resource/Uniform\_Resource\_Identifier">http://yago-knowledge.org/resource/Uniform\_Resource\_Identifier</a>	
Meridian	dbpedia	<a href="http://dbpedia.org/resource/Meridian\_(geography)">http://dbpedia.org/resource/Meridian\_(geography)</a>	0.719119
	freebase	<a href="http://rdf.freebase.com/ns/m.02yd57">rdf.freebase.com/ns/m.02yd57</a>	

Tabla 4.6: Conceptos más importantes según AlchemyApi

### Conclusiones del método de búsqueda por conceptos

Seleccionar los conceptos importantes dentro de un texto puede ayudar a conocer su temática y si están enlazados con Linked Data asegura su existencia dentro de algún esquema. Sin embargo, el problema de realizar la selección de los conceptos de forma automática, es que los conceptos apuntan a palabras demasiado específicas como se muestran en los resultados de la tabla 4.6, mientras la etiquetación en los *datasets* dentro del repositorio trata sobre conceptos generales como (geografía, comida, medicina, etc.).

#### 4.2.3.4. Búsqueda por extracción de entidades

Una forma de reducir el número de términos (Cómo se plantea en 4.2.3.1) a ser usados en la búsqueda sobre un repositorio consiste en obtener una clasificación o dominio para dichos términos. Para obtener el dominio se planteó la utilización del reconocimiento de entidades, con ayuda de herramientas como AlchemyApi y GATE(ANNIE), puesto que de las entidades obtenidas se puede obtener una clasificación. Las categorías que se asignan a las entidades son una generalización de los conceptos que representan dichas entidades, por ejemplo dos entidades “*City*” y “*Country*” se agrupan bajo la misma clasificación: ”*Location*”. De esta manera se agrupan los términos de la entrada de texto bajo conceptos fijos y de una mayor abstracción, con lo cual se puede obtener información sobre el contexto del que trata la entrada de texto, mediante el análisis de las categorías de las entidades extraídas.

Para la implementación de esta técnica de extracción de información como mecanismo de filtrado de términos de búsqueda, dentro de la selección de ontologías en los repositorios disponibles en Internet, se consideró lo siguiente:

- La entrada de texto constó de los nombres de los parámetros enriquecidos (sinónimos y sugerencias) pertenecientes al Servicio Web para el cual se busca una ontología.
- Para el ejemplo se usó el servicio Eventful, la lista de términos usados como entrada de texto se puede ver en la tabla 4.1.
- Se usaron dos herramientas de extracción de entidades: AlchemyAPI y GATE(ANNIE). Esto debido a que estas dos herramientas son las más conocidas y usadas actualmente.

### Ejecución de las pruebas usando GATE

ANNIE (Sub proyecto de GATE) se encuentra pre configurada para analizar textos en inglés, provee extracción de entidades basada la clasificación original de la MUC<sup>21</sup>, la cual se puede ver en más de detalle en la tabla 4.7.

Tipo de entidad	Descripción
Person	Referencias a personas tanto de sexo masculino como femenino.
Location	Lugares como: Ciudades, países, aeropuertos, etc.
Money	Referencias a cantidades de dinero.
Percent	Porcentajes
Date	Referencias temporales.
Address	Emails, números telefónicos, IP's, etc.
Lookup	Se puede encontrar en las lista de entidades comunes, pero no se puede definir un tipo de entidad.
Unknown	Se reconoce como un nombre propio, pero no se puede determinar un tipo de entidad.

Tabla 4.7: Tipos de entidades soportadas por GATE

Para la ejecución de esta prueba se utilizó GATE Developer<sup>22</sup>, una herramienta que permite ejecutar los procesos de ANNIE desde un ambiente gráfico. Esta herramienta permite visualizar mediante un esquema de colores las entidades obtenidas del texto con sus clasificaciones. El resultado de la prueba visualizado sobre GATE Developer se muestra en la figura 4.1.

### Resultados obtenidos con GATE

El resultado obtenido de la prueba se muestra en la tabla 4.8, la cual indica las entidades que fueron reconocidas y su tipo.

<sup>21</sup>Message Understanding Conference

<sup>22</sup><https://gate.ac.uk/family/developer.html>

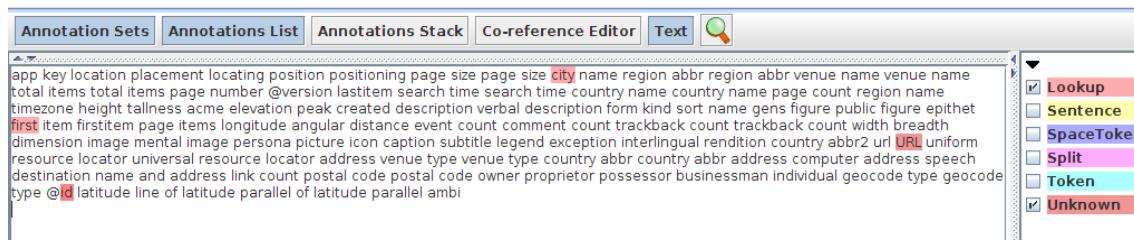


Figura 4.1: Resultados sobre GATE Developer

Entidad	Tipo
City	Lookup
First	Lookup
URL	Unknown
Id	Lookup

Tabla 4.8: Entidades obtenidas para el servicio Eventful usando GATE

### Ejecución de las pruebas usando AlchemyApi

AlchemyApi tiene soporte para el idioma inglés y tiene una lista completa de tipos de entidades que puede reconocer. AlchemyApi puede reconocer 41 tipos de entidades y 975 sub tipos, la lista completa se puede encontrar en la documentación del AlchemyApi<sup>23</sup>. Además, esta herramienta provee ponderación de entidades de acuerdo a su relevancia dentro del texto y las asocia con conceptos de Linked Data. Para la ejecución de las pruebas se usó la demo online<sup>24</sup> al igual que se hizo en las secciones 4.2.3.2, 4.2.3.1 y 4.2.3.3.

Los resultados obtenidos de la extracción de entidades se muestran en la tabla 4.9.

Entidad	Tipo
Id	TwitterHandle
Version	TwitterHandle

Tabla 4.9: Resultados para el servicio Eventful usando AlchemyApi

<sup>23</sup><http://go.alchemyapi.com/hs-fs/hub/396516/file-2576982340-xls/AlchemyAPI-EntityTypesandSubtypes.xls>

<sup>24</sup><http://www.alchemyapi.com/products/demo/alchemylanguage>

## Conclusiones para la búsqueda con la extracción de entidades

La reducción de los términos a usar para la búsqueda de ontologías a través de la extracción de entidades y más específicamente la utilización de las categorías de las entidades como términos de búsqueda no es aplicable para este componente. Las implementaciones de NER existentes utilizan técnicas lingüísticas basadas en la gramática del lenguaje y modelos probabilísticos [50], las cuales sólo funcionan con textos descriptivos que sigan reglas gramaticales. Estos algoritmos de extracción de entidades no funcionan sobre simples listas de palabras que no cumplen las reglas gramaticales de un lenguaje, como es el caso de la lista de parámetros de un servicio. Es por esta razón que ni AlchemyApi ni ANNIE proveen entidades relevantes a partir de la lista de parámetros del Servicio Web (entrada del componente), lo que se evidencia en los resultados de las pruebas realizadas(tablas 4.8 y 4.9). El reducido número de entidades obtenidas indican que esta técnica no provee los resultados necesarios para realizar una búsqueda sobre los repositorios de ontologías y por tanto no podría ser utilizada para el componente de selección de ontologías.

### 4.2.4. Búsqueda por dominio, usando medidas de relación semántica

Una alternativa al uso de AlchemyApi para la detección del dominio de un Servicio Web es medir la relación entre los nombres de los parámetros del Servicio Web con los nombres de los dominios seleccionados. Actualmente existen bases de datos léxicas como WordNet<sup>25</sup> o Web Dante<sup>26</sup> las cuales almacenan las palabras de un idioma (Inglés para este caso) con sus respectivos tipos (Verbos, Sustantivos, Adjetivos, etc.), además de guardar relaciones entre las palabras como: sinónimos, meronimia, antónimos, etc ([51]). Usando estas bases de datos se han desarrollado una serie de medidas de relación entre términos, las cuales permiten cuantificar el nivel de relación entre dos términos de un lenguaje [63]. Estas medidas de relación pueden ser aplicadas en la detección del dominio en un servicio, mediante la selección del dominio que retorne el mayor valor de relación promedio entre los nombres de los parámetros del servicio y el nombre del dominio. Este enfoque ha sido probado exitosamente en la clasificación de Servicios Web SOAP dentro del proyecto SAWSSDL Generator[11]

---

<sup>25</sup><https://wordnet.princeton.edu/>

<sup>26</sup><http://www.webdante.com/>

Existen siete métricas de similitud semántica principales que se utilizan dentro de la comunidad científica[26], una implementación de estos algoritmos sobre la base de datos WordNet se puede encontrar en el proyecto WS4J<sup>27</sup>. A través de esta implementación se puede aplicar todas estas medidas de forma transparente y rápida sin entrar en detalles de implementación. Sin embargo, las pruebas realizadas en el estudio SAWSDL Generator [11] demuestran que las medidas más adecuadas para la clasificación de Servicios Web dentro un dominio son: LIN y JIANG

### Ejecución de las pruebas

Para la ejecución de las pruebas se realizó las siguientes consideraciones:

- Se usó el Servicio Web Eventful para la realización de esta prueba como en los demás casos (Tabla 4.1).
- Dado que se necesita la elección de una lista de dominios en los cuales los Servicios Web se puedan agrupar, se usó la clasificación definida por LOD (sección 4.2.3.1) para la realización de esta prueba.
- Se utilizó la implantación de los algoritmos de la demo oficial del proyecto WS4J<sup>28</sup>.
- Se utilizaron los algoritmos LIN y JIANG.
- Se consideró como la clasificación más acertada aquella que contiene el mayor puntaje, basado en el promedio de los valores obtenidos con cada algoritmo para cada palabra comparada.

Los resultados de las pruebas se pueden ver en la tabla 4.10. De acuerdo a este método y usando la lista de dominios definidos en LOD Cloud, se debería buscar dentro de Datahub ontologías en las categorías: Goverment y Publications. Sin embargo, al buscar ontologías con SPARQL Endpoints dentro de estas categorías no se obtuvieron resultados.

---

<sup>27</sup><https://code.google.com/p/ws4j/>

<sup>28</sup><http://ws4jdemo.appspot.com/>

Termino	Multimedia		Geography		Government		Publications		Life Sciences	
	LIN	JING	LIN	JING	LIN	JING	LIN	JING	LIN	JING
app	-	-	-	-	-	-	-	-	-	-
key	-	-	-	-	-	-	-	-	-	-
location	0.00	0.00	0.00	0.00	0.2842	0.1065	0.2691	0.1392	0.00	0.00
placement	0.00	0.00	0.00	0.00	0.2842	0.0866	0.2691	0.0755	0.00	0.00
locating	-	-	-	-	-	-	-	-	-	-
position	0.00	0.00	0.00	0.00	0.3158	0.0923	0.2892	0.0887	0.00	0.00
positioning	-	-	-	-	-	-	-	-	-	-
page	0.00	0.00	0.00	0.00	0.1068	0.0767	0.0849	0.0741	0.00	0.00
size	0.00	0.00	0.00	0.00	0.1135	0.0822	0.0891	0.0721	0.00	0.00
city	0.00	0.00	0.00	0.00	0.3887	0.0901	0.1946	0.1034	0.00	0.00
name	0.00	0.00	0.00	0.00	0.5862	0.1157	0.4128	0.0777	0.00	0.00
region	0.00	0.00	0.00	0.00	0.00	0.0999	0.2306	0.1282	0.00	0.00
abbr	-	-	-	-	-	-	-	-	-	-
venue	0.00	0.00	0.00	0.00	0.00	0.0573	0.1330	0.0656	0.00	0.00
total	-	-	-	-	-	-	-	-	-	-
items	0.00	0.00	0.00	0.00	0.3130	0.0821	0.2007	0.0822	0.00	0.00
number	0.00	0.00	0.00	0.00	0.3887	0.0991	0.7815	0.2745	0.00	0.00
version	0.00	0.00	0.00	0.00	0.3491	0.0790	0.1831	0.0633	0.00	0.00
lastitem	-	-	-	-	-	-	-	-	-	-
search	0.00	0.00	0.00	0.00	0.2918	0.0791	0.2759	0.0732	0.00	0.00
time	0.00	0.00	0.0001	0.00	0.3165	0.0926	0.2728	0.0832	0.00	0.00
country	0.00	0.00	0.0002	0.00	0.5874	0.1828	0.1974	0.1052	0.00	0.00
count	0.00	0.00	0.0003	0.00	0.2749	0.0728	0.2608	0.0677	0.00	0.00
timezone	-	-	-	-	-	-	-	-	-	-
height	0.00	0.00	0.0005	0.00	0.0995	0.0709	0.0802	0.0633	0.00	0.00
tallness	0.00	0.00	0.0006	0.00	0.0990	0.0705	0.0798	0.0630	0.00	0.00
acme	0.00	0.00	0.0007	0.00	0.0984	0.0700	0.1384	0.0687	0.00	0.00
elevation	0.00	0.00	0.0008	0.00	0.2493	0.0726	0.2376	0.0865	0.00	0.00
peak	0.00	0.00	0.0009	0.00	0.1052	0.0754	0.1508	0.0759	0.00	0.00
created	-	-	-	-	-	-	-	-	-	-
description	0.00	0.00	0.0011	0.00	0.2802	0.0772	0.2655	0.0694	0.00	0.00
verbal	-	-	-	-	-	-	-	-	-	-
form	0.00	0.00	0.0013	0.00	0.4783	0.1300	0.2248	0.0902	0.00	0.00
kind	0.00	0.00	0.0014	0.00	0.4284	0.1105	0.2248	0.0820	0.00	0.00
sort	0.00	0.00	0.0015	0.00	0.4284	0.1105	0.2248	0.0820	0.00	0.00
gens	-	-	-	-	-	-	-	-	-	-
figure	0.00	0.00	0.0017	0.00	0.1243	0.0910	0.6305	0.1752	0.00	0.00
public	-	-	-	-	-	-	-	-	-	-
epithet	0.00	0.00	0.0019	0.00	0.0914	0.0646	0.0749	0.0582	0.00	0.00
first	-	-	-	-	-	-	-	-	-	-
item	0.00	0.00	0.0021	0.00	0.3130	0.0821	0.2007	0.0822	0.00	0.00
firstitem	-	-	-	-	-	-	-	-	-	-
longitude	-	-	-	-	-	-	-	-	-	-
angular	-	-	-	-	-	-	-	-	-	-
distance	0.00	0.00	0.0025	0.00	0.1161	0.0842	0.1520	0.0766	0.00	0.00
event	0.00	0.00	0.0026	0.00	0.3961	0.1455	0.3633	0.1265	0.00	0.00
comment	0.00	0.00	0.0027	0.00	0.2926	0.0829	0.2766	0.0734	0.00	0.00
trackback	-	-	-	-	-	-	-	-	-	-
width	-	-	-	-	-	-	-	-	-	-
breadth	0.00	0.00	0.0030	0.00	0.4491	0.0795	0.1677	0.0612	0.00	0.00
dimension	0.00	0.00	0.0031	0.00	0.4003	0.0984	0.2100	0.0751	0.00	0.00
image	0.00	0.00	0.0032	0.00	0.3823	0.0912	0.6849	0.2232	0.00	0.00
mental	-	-	-	-	-	-	-	-	-	-
persona	0.00	0.00	0.0034	0.00	0.6413	0.1535	0.2868	0.0772	0.00	0.00
picture	0.00	0.00	0.0035	0.00	0.3620	0.0845	0.8239	0.4138	0.00	0.00
icon	0.00	0.00	0.0036	0.00	0.00	0.0710	0.6849	0.2232	0.00	0.00
caption	0.00	0.00	0.0037	0.00	0.00	0.00	0.00	0.00	0.00	0.00
subtitle	0.00	0.00	0.0038	0.00	0.00	0.00	0.00	0.00	0.00	0.00
legend	0.00	0.00	0.0039	0.00	0.1010	0.0720	0.0811	0.0642	0.00	0.00
exception	0.00	0.00	0.0040	0.00	0.2917	0.0742	0.1870	0.0650	0.00	0.00
Totales	0.00	0.00	0.0581	0	<b>10.8287</b>	3.557	10.5956	<b>4.0498</b>	0	0

Tabla 4.10: Resultados de la clasificación del Servicio Web Eventful

## Conclusiones para la búsqueda con medidas de relación semántica

Este método de clasificación puede ser aplicado en la definición de dominios para Servicios Web como se demuestra en la publicación SAWSDL [11]. Además, los resultados obtenidos en este experimento pueden ser mejorados con el uso de un listado de dominios más amplio y el filtrado de términos antes de ejecutar los algoritmos. Sin embargo, este método tiene un inconveniente y es que para su pleno funcionamiento requiere que las ontologías sean previamente clasificadas dentro de los dominios que se definan. En el caso de Datahub la búsqueda se realiza por el título de la ontología y las etiquetas ingresadas por un usuario al momento de publicar la ontología. Esto dificulta la búsqueda puesto que no se puede encontrar fácilmente ontologías para un dominio dado, ya que en la mayoría de casos el publicador no provee una descripción adecuada de la ontología. Este método sería aplicable si las ontologías publicadas en los repositorios (Datahub) estuvieran correctamente etiquetadas en el dominio al que pertenecen. Un problema adicional que se identificó es que las ontologías *Cross-Domain*<sup>29</sup> no podrían ser clasificadas con este método.

### 4.3. Buscadores semánticos

Los motores de búsqueda semánticos como se trató en la sección 2.5.1 son los equivalentes a los motores de búsqueda de la Web normal, pero aplicados a recursos semánticos. Estos motores debido a la facilidad de manejo y a que permiten encontrar ontologías conociendo para esto solo conceptos relacionados, han ganado bastante popularidad. Existen varios buscadores semánticos actualmente, algunos de los cuales están orientados a buscar conceptos, mientras otros proveen ontologías y unos cuantos proveen ambas funciones. En la tabla 4.11 se presenta un análisis de buscadores semánticos en los cuales se observaron las características destacadas que poseen cada uno. El análisis de los buscadores se realizó con el fin de determinar cuáles son los más propicios dependiendo de sus capacidades para poder pertenecer al componente de búsqueda y selección de ontologías dentro del proceso de anotación. Luego de analizar las características destacadas de cada buscador semántico y basado en la disponibilidad del recurso, en la facilidad de uso, y la información que proveen se ha seleccionado dos candidatos, los cuales serán descritos a continuación:

---

<sup>29</sup>Ontología conformada por múltiples dominios

Buscador	Habilitado	Características	Enlace
Aquabrowser	No desde Enero 2015	Buscador de libros y artículos. Gráfico de conceptos semánticos relacionados	<a href="http://aquabrowser.lib.ed.ac.uk/">http://aquabrowser.lib.ed.ac.uk/</a>
Aqualog	Si. Demo no disponible actualmente	Prototipo para responder preguntas de lenguaje natural basadas en ontologías	<a href="http://technologies.kmi.open.ac.uk/aqualog/">http://technologies.kmi.open.ac.uk/aqualog/</a>
Autofocus	Si. No posee demo	Solución privativa para organización rdfs	<a href="http://www.aduna-software.com/">http://www.aduna-software.com/</a>
BrowseRDF	Si. Descarga disponible	Buscador de RDF de forma local. Lenguaje Ruby	<a href="https://launchpad.net/browserdf">https://launchpad.net/browserdf</a>
DBin	Si. Demo disponible	Creación de grupos y etiquetado para formar rdf y owl . Intercambio P2P.	<a href="http://dbin.org/">http://dbin.org/</a>
DBpedia	No. Solo página de información	Buscador al igual que Wikipedia que hace uso de información semántica (RDFs)	<a href="http://wiki.dbpedia.org/FacetedSearch">http://wiki.dbpedia.org/FacetedSearch</a>
e-culture	Si. Buscador disponible	Búsquedas por palabras, resultados a ámbitos limitados (arte).	<a href="http://e-culture.multimedian.nl/demo/session/search">http://e-culture.multimedian.nl/demo/session/search</a>
Falcons	Si. Buscador disponible	Buscador por palabras. Permite buscar objetos, conceptos y ontologías. Presenta las ontología en gráficos. API	<a href="http://ws.nju.edu.cn/falcons">http://ws.nju.edu.cn/falcons</a>
FLink	Si. Buscador disponible	Buscador para uso biomédico.	<a href="http://www.ncbi.nlm.nih.gov/Structure/flink/docs/flink_how_to.html">http://www.ncbi.nlm.nih.gov/Structure/flink/docs/flink_how_to.html</a>
FreeBase	Si. Buscador disponible	Búsquedas por palabra especificando sentido semántico . Ofrece propiedades relacionadas, descripción.	<a href="https://www.firebaseio.com/">https://www.firebaseio.com/</a>
Ginseng	Si. Solo información	Permite realizar consultas en lenguaje natural las cuales son traducidas a SPARQL	<a href="https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/talking-to-the-semantic-web/ginseng/">https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/talking-to-the-semantic-web/ginseng/</a>
H-DOSE	Si. Descarga disponible	Administración de información ontológica, indexación y recuperación de ontologías.	<a href="http://elite.polito.it/index.php/research/research-topics/59-past-research-projects/227-h-dose?showall=1">http://elite.polito.it/index.php/research/research-topics/59-past-research-projects/227-h-dose?showall=1</a>
Haystack	Si. Descarga disponible	Permite crear, administrar, compartir información semántica. Crear buscadores .Usa Django y python.	<a href="http://www.w3.org/2005/04/swls/BioDash/Demo/WhatIsHaystack.html">http://www.w3.org/2005/04/swls/BioDash/Demo/WhatIsHaystack.html</a>
hybrid-search	Si. Solo información	Buscador de elementos como documentos, carpetas	<a href="http://www.bhybrid.com/en/h/a6/Hybrid-Search">http://www.bhybrid.com/en/h/a6/Hybrid-Search</a>
KIM	Solo información	Análisis de texto. Plataforma que ayuda a crear enlaces y anotaciones semánticas . Provee buscadores	<a href="http://www.ontotext.com/products/ontotext-semantic-platform/">http://www.ontotext.com/products/ontotext-semantic-platform/</a>
Longwell	No disponible	Buscador de rdfs con aspectos	
mSpace	Si. Demo disponible	Buscador con facetas	<a href="http://mspace.fm/mspace/">http://mspace.fm/mspace/</a>
OntoKhoj	Solo publicación	Clasificador y buscador de ontologías	<a href="http://dl.acm.org/citation.cfm?id=956712">http://dl.acm.org/citation.cfm?id=956712</a>
Ontosearch	Si. Demo disponible aunque no funcional	Buscador de instancias, identifica el tipo	<a href="http://www.ontosearch.com/">http://www.ontosearch.com/</a>
OntoWiki	Si. Demo disponible con clave	Ayuda a la representación visual de contenido en forma de mapa	<a href="http://aksweb.org/Projects/OntoWiki.html">http://aksweb.org/Projects/OntoWiki.html</a>
Openacademia	No disponible	Buscador	
OWLIR	Solo publicación	Buscador con inferencia. Año 2003	<a href="http://citeserx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.4082&amp;rep=rep1&amp;type=pdf">http://citeserx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.4082&amp;rep=rep1&amp;type=pdf</a>
QuizRDF	No encontrado	Buscador que trata de combinar capacidades de searching y browsing. Búsqueda específica y exploratoria	
SemSearch	Si. Descarga disponible	Buscador por palabras claves. Utiliza Apache Lucence	<a href="http://technologies.kmi.open.ac.uk/semsrch/">http://technologies.kmi.open.ac.uk/semsrch/</a>
SHOE	Si. Demo no funcional	Búsqueda mediante etiquetas XML .Utiliza inteligencia artificial para acelerar búsquedas.	<a href="http://www.cs.umd.edu/projects/plus/SHOE/search/#instruct">http://www.cs.umd.edu/projects/plus/SHOE/search/#instruct</a>
SLASHFACET	Si. Demo no disponible actualmente	Trabaja sobre RDFS, permite autocompletar dentro de las búsquedas.	<a href="http://slashfacet.semanticweb.org/">http://slashfacet.semanticweb.org/</a>
Squiggle	Si. Demo disponible	búsqueda con características sintácticas y semánticas. Buscador semántico para un dominio específico. Utilizar Sesame y Lucene	<a href="http://swa.cefriel.it/Squiggle">http://swa.cefriel.it/Squiggle</a>
SWSE	Si. Demo no disponible actualmente	Buscador con técnicas de inferencia.	<a href="http://www.swse.org">http://www.swse.org</a>
TAP	Si. Demo no disponible actualmente	Permite ver términos de una base de conocimiento. Determina el tipo de datos .Proyecto año 2003	<a href="http://www.w3.org/2002/05/tap/">http://www.w3.org/2002/05/tap/</a>
Topia	Solo información	presentación de estructuras sobre objetos multimedia a través de consultas	<a href="http://www.cwi.nl/Topia">http://www.cwi.nl/Topia</a>
SWISE	Solo información	Buscador que mejora resultados con inteligencia artificial	Paper IEEE : SWISE: Semantic Web based intelligent search engine
Sindice	Si. Buscador disponible	Buscador de documentos semánticos. Simple y con filtros avanzados.	<a href="http://sindice.com/">http://sindice.com/</a>
Watson	Si. Buscador disponible	Permite encontrar URIs de recursos como conceptos, clases, propiedades a través de keywords.	<a href="http://watson.kmi.open.ac.uk/Overview.html">http://watson.kmi.open.ac.uk/Overview.html</a>

Tabla 4.11: Características de Buscadores semánticos

### 4.3.1. WATSON

Watson<sup>30</sup> es un motor de búsqueda semántico que permite encontrar ontologías, conceptos y documentos semánticos utilizando palabras claves como entradas [47]. Los resultados que brinda esta herramienta al realizar la búsqueda son las URIs de los documentos semánticos o de los conceptos, en donde las palabras claves aparecen como identificadores o literales de clases, propiedades e individuales. Dentro de los resultados, además se puede encontrar información adicional, que brinda nuevas funcionalidades relacionadas con el recurso como: conocer comentarios asociados, gráficos, acceder a lugares en los que encuentra el recurso, acceso a SPARQL Endpoint, etc .

Para el funcionamiento del motor de búsqueda, Watson realiza 3 actividades principales [47]:

1. Recolectar los datos semánticos disponibles en la Web.
2. Analizar y extraer metadatos útiles e índices.
3. Implementar consultas eficientes para acceder a los datos.

Aunque las tareas indicadas son parte de la mayoría de motores de búsqueda ontológicos, la implementación basada en un conjunto de componentes dista un poco de las demás por las tecnologías que involucra y el trato que tiene la información semántica. Algunas de las tecnologías que forma parte de los componentes de Watson son:

- Para el descubrimiento de las localizaciones de los recursos semánticos la herramienta Heritrix<sup>31</sup>, que forma parte del componente de *crawling* y rastreo.
- Para la generación de índices eficientes, el sistema de indexación que se utiliza en Apache Lucene.

#### API de Watson

Watson provee muchas de sus funcionalidades a través de un conjunto de servicios que se puede acceder mediante un API en formato REST o SOAP. Cada uno de los servicios da como resultado un conjunto específico de datos los cuales

---

<sup>30</sup><http://watson.kmi.open.ac.uk/WatsonWUI/>

<sup>31</sup><http://crawler.archive.org/>

pueden estar en formato XML, JSON o Texto plano. Algunos de los principales funcionalidades que se puede acceder a través de los diferentes servicios son:

- Búsqueda de documentos semánticos
- Obtención de metadatos de los documentos semánticos.
- Localización de entidades en documentos semánticos
- Búsqueda de entidades dentro de documentos semánticos específicos.
- Obtención de etiquetas de una entidad
- Relaciones de y hacia una entidad

### Pruebas sobre Watson

Para verificar la funcionalidad que pueda tener Watson sobre el componente de búsqueda y selección de ontologías se ha realizado un conjunto de pruebas, utilizando la API REST del buscador. El objetivo de las pruebas fue encontrar el acceso a una ontología, para lo cual se realizó consultas al servicio de búsqueda de documentos semánticos del API de Watson, utilizando como entrada el conjunto de palabras obtenido del servicio Eventful listado en la tabla 4.1.

Los resultado de la pruebas como se puede observar en el segmento de código de respuesta 4.3, es vacío, lo que significa que las ontologías del buscador deben contener todas las palabras ingresadas para ser mostrados como resultados. La necesidad de que todas las palabras clave estén presentes en los resultados es un problema recurrente en los buscadores, demostrando que ingresar todo el conjunto de *keywords* del servicio no es una opción viable.

<sup>1</sup><SemanticContent–array />

Segmento de Código 4.3: Respuesta Watson prueba con todas las palabras

Debido a que un número grande de palabras anula los resultados de la búsqueda, se disminuyó en las pruebas a un número reducido de palabras seleccionadas. En la tabla 4.12 se muestra algunas combinaciones de entradas con un número reducido de palabras y los resultados que se obtuvieron con el servicio.

Palabras clave	Num. Resultados	URI resultantes (5 primeras)	Localización documento	Recurso encontrado
location position event address timezone placement	3	<a href="http://ontologyportal.org/translations/SUMO.owl.txt">http://ontologyportal.org/translations/SUMO.owl.txt</a> <a href="http://reliant.teknowledge.com/DAML/SUMO.daml">http://reliant.teknowledge.com/DAML/SUMO.daml</a> <a href="http://reliant.teknowledge.com/DAML/SUMO.owl">http://reliant.teknowledge.com/DAML/SUMO.owl</a>	<a href="http://ontologyportal.org/translations/SUMO.owl.txt">http://ontologyportal.org/translations/SUMO.owl.txt</a> <a href="http://reliant.teknowledge.com/DAML/SUMO.daml">http://reliant.teknowledge.com/DAML/SUMO.daml</a> <a href="http://reliant.teknowledge.com/DAML/SUMO.owl">http://reliant.teknowledge.com/DAML/SUMO.owl</a>	Página no disponible No disponible No disponible
location event address	200	<a href="http://www.mindswap.org/2004/terrorOnt.owl">http://www.mindswap.org/2004/terrorOnt.owl</a> <a href="http://daml.umbc.edu/ontologies/ittalks/event">http://daml.umbc.edu/ontologies/ittalks/event</a> <a href="http://www.aktors.org/ontology/portal.daml">http://www.aktors.org/ontology/portal.daml</a> <a href="http://www.mindswap.org/2003/owl/swint/terrorism">http://www.mindswap.org/2003/owl/swint/terrorism</a> <a href="http://oeai.ontologymatching.org/2004/Contest/204/onto.rdf">http://oeai.ontologymatching.org/2004/Contest/204/onto.rdf</a>	<a href="http://www.mindswap.org/2004/terrorOnt.owl">http://www.mindswap.org/2004/terrorOnt.owl</a> <a href="http://daml.umbc.edu/ontologies/ittalks/event">http://daml.umbc.edu/ontologies/ittalks/event</a> <a href="http://www.aktors.org/ontology/portal.daml">http://www.aktors.org/ontology/portal.daml</a> <a href="http://www.mindswap.org/2003/owl/swint/terrorism">http://www.mindswap.org/2003/owl/swint/terrorism</a> <a href="http://oeai.ontologymatching.org/2004/Contest/204/onto.rdf">http://oeai.ontologymatching.org/2004/Contest/204/onto.rdf</a>	Página Web OWL Página no disponible Página Web RDF
location event	622	<a href="http://ebiquity.umbc.edu/ontology/event.owl#event">http://ebiquity.umbc.edu/ontology/event.owl#event</a> <a href="http://www.mindswap.org/2004/terrorOnt.owl">http://www.mindswap.org/2004/terrorOnt.owl</a> <a href="http://www.isi.edu/webscripter/event.o.daml">http://www.isi.edu/webscripter/event.o.daml</a> <a href="http://www.daml.org/2001/12/charter/event-ont">http://www.daml.org/2001/12/charter/event-ont</a> <a href="http://www.daml.org/2002/01/coabs-daml/event-ont">http://www.daml.org/2002/01/coabs-daml/event-ont</a>	<a href="http://ebiquity.umbc.edu/ontology/event.owl">http://ebiquity.umbc.edu/ontology/event.owl</a> <a href="http://www.mindswap.org/2004/terrorOnt.owl">http://www.mindswap.org/2004/terrorOnt.owl</a> <a href="http://www.isi.edu/webscripter/event.o.daml">http://www.isi.edu/webscripter/event.o.daml</a> <a href="http://www.daml.org/2001/12/charter/event-ont.owl">http://www.daml.org/2001/12/charter/event-ont.owl</a> <a href="http://www.daml.org/2002/01/coabs-daml/event-ont">http://www.daml.org/2002/01/coabs-daml/event-ont</a>	OWL Página Web RDF OWL OWL

Tabla 4.12: Pruebas Watson

### Conclusiones del análisis de Watson

Watson es un buscador ontológico muy completo, pensado para ofrecer funcionalidades tanto para el acceso para usuarios normales a través de su intuitiva página Web, así como para desarrolladores por medio de su API. En el caso del acceso para desarrolladores, Watson fue diseñado con la intención de formar parte de proyectos más grandes relacionados con la Web semántica como describe su documentación. Por esta razón Watson no limita sus funcionalidades a las de un simple buscador, sino que añade características apreciables dentro una herramienta como acceso a metadatos, descripciones de los recursos, revisiones, estadísticas de los elementos encontrados, entre otras. Algunas de estas características pueden ser de especial interés para el componente de selección de ontologías principalmente al acceso a SPARQL Endpoint, registro de las ubicaciones en los que se encuentran los recursos ontológicos y especificación de los lenguajes de representación (RDF, OWL, DAML, etc). Sin embargo, aunque Watson se muestra

como un buen candidato según la documentación disponible, presenta algunos problemas en la práctica que dificultan su utilización. Uno de estos problemas se debe a la disponibilidad de muchos elementos que aunque constan dentro de la documentación, ya no se encuentran funcionales. Entre problemas encontrados están:

- Acceso a SPARQL Endpoint temporalmente no disponible. (Según informe de error<sup>32</sup>)
- Servicios de acceso a metadatos y obtención de etiquetas con problemas.
- Localización de recursos hacia sitios donde no los contienen.

Problemas que dificultan la obtención de las ontologías a través de los documentos encontrados. A esto se suman otros problemas que forman parte de las complicaciones presentadas por el buscador, como documentos semánticos resultantes con pocos elementos que no soportarían al gran número de propiedades con los que se tiene que comparar y un problema recurrente entre los métodos probados, que es la disminución drástica de resultados frente a un gran número de entradas.

#### 4.3.2. Falcons

Falcons<sup>33</sup> es un motor de búsqueda semántico basado en palabras claves. Como resultados a las búsquedas permite obtener URIs a los objetos y conceptos pertenecientes a la Web Semántica que coinciden con las palabras ingresadas. Los resultados para facilitar su interpretación son resumidos por cada entidad (objetos, clases, propiedades) mostrando además el número total de elementos encontrados de ese tipo. Adicionalmente dependiendo del tipo de entidad se puede acceder a información relacionada. Falcons destaca por la administración interna de su información a través de documentos virtuales que relaciona todos los elementos encontrados como un documento semántico que facilitan la extracción de su información [27]. Para extraer ontologías, Falcons también ofrece una opción, que relaciona las clases y propiedades de los documentos virtuales que en si representan ontologías con las palabras ingresadas. Los resultados a las búsquedas de ontología se realizan en función del número de coincidencias con los elementos y su popularidad (enlaces con otros recursos semánticos). Para facilitar la

---

<sup>32</sup>Pruebas realizadas el día: 09-12-2014

<sup>33</sup><http://ws.nju.edu.cn/falcons/objectsearch/index.jsp>

visualización de resultados de ontologías se muestran los elementos y sus relaciones en un gráfico resumido. Otra de las características destacadas por Falcons es que presenta como información adicional a los resultados con cada ontología encontrada un conjunto de ontologías relacionadas. Las ontologías relacionadas pueden ser mostradas dependiendo del nivel de relación que se desea, siendo muy fuerte o débil según un conjunto de medidas como : relación semántica, similitud de contenido, expresividad, relación de distribución. Las ontologías además guardan un conjunto de metadatos para facilitar su interpretación por las personas, aunque se limitan a datos básicos como título, descripción, publicador y fecha de modificación.

### API Falcons

Falcons provee su funcionalidad para desarrolladores a través de Servicios Web REST. Estos deben estar codificados en UTF-8<sup>34</sup> para su correcto funcionamiento. A continuación se describe los servicios disponibles para la búsqueda que pone a disposición de los usuarios y desarrolladores.

- Servicios de búsqueda de objetos
- Servicio de búsqueda de clases
- Servicio de búsqueda de propiedades
- Servicio de búsqueda de documentos.

No está disponible un servicio para búsquedas de ontologías, por lo que no se puede acceder a esta funcionalidad mediante el API.

### Pruebas de Falcons

Se realizaron un conjunto de pruebas de forma análoga a las realizadas en Watson utilizando el motor de búsqueda Falcons para verificar su factibilidad para su uso dentro del componente de búsqueda y selección de ontologías. Para la prueba se utilizó la búsqueda de documentos semánticos a través de la página. No se utilizó el API para este caso debido a que este solo recibe como parámetro una URI de identificación recurso y se dispone únicamente del conjunto de palabras de entrada utilizados en prueba anteriores (tabla 4.1).

---

<sup>34</sup><http://www.utf8-chartable.de/>

```
Your search - app_key location placement positioning page_size pagesize city_name region_abbr regionabbr venue_name venuname total_items totalitems page_number @version last_item
search_time searchtime country_name countryname page_count region_name timezone height tallness acme elevation peak created description verbadescription form kind sort name gens figure
publicfigure epilhet first_item firstitem page_items longitude angulardistance event_count comment_count trackback_count trackbackcount width breadth dimension image mentalimage persona picture icon
caption subtitle legend exception interlingualrendition country_abbr2 url URL uniformresourcelocator universalresourcelocator address venue_type venuetype country_abbr countryabbr address
computeraddress speech destination nameandaddress link_count postal_code postalcode owner proprietor possessor businessman individual geocode_type geocodetype @id latitude lineoflatitude
paralleloflatitude parallel_ambit - did not match any documents.
```

Suggestions:

- Make sure all words are spelled correctly.
- Try different or less keywords.

Figura 4.2: Resultado obtenido con Falcons

Luego de ingresar todo el conjunto de palabras en el buscador no se obtuvo documentos como resultado, figura 4.2. En lugar a los resultados esperados, se obtuvo un mensaje mostrando que no existen documentos que coincidan con las entradas. Como sugerencia propone revisar que las palabras estén escritas correctamente, usar palabras diferentes o ingresar un menor número palabras.

Dado que para un gran número de palabras no se encontraron resultados, siguiendo el estilo de las pruebas en los otros métodos se usó un número reducido de palabras seleccionadas. Las pruebas se pueden ver en la tabla 4.13.

Palabras clave	Num. Resultados	Nombre documento	Localización documento	Recurso encontrado
location position event address timezone placement	0			
location event address	531	<a href="http://www.ilrt.bris.ac.uk/discovery/2001/06/content/sws2001-07-30.rdf">http://www.ilrt.bris.ac.uk/discovery/2001/06/content/sws2001-07-30.rdf</a> AKT Reference Ontology (Portal Ontology) RDF : Alexandre Alapetite VHL GENE; VHL vida. en todas sus manifestaciones (Ruso)	<a href="http://www.ilrt.bris.ac.uk/discovery/2001/06/content/sws2001-07-30.rdf">http://www.ilrt.bris.ac.uk/discovery/2001/06/content/sws2001-07-30.rdf</a> <a href="http://www.aktors.org/ontology/portal">http://www.aktors.org/ontology/portal</a> <a href="http://alexandre.alapetite.net/cv/foaf.rdf.xml">http://alexandre.alapetite.net/cv/foaf.rdf.xml</a> <a href="http://bio2rdf.org/omim:608537">http://bio2rdf.org/omim:608537</a> <a href="http://www.liveinternet.ru/users/surd_event/foaf/">http://www.liveinternet.ru/users/surd_event/foaf/</a>	RDF Página no disponible Página de descripción Página de descripción Página no disponible
location event	5043	<a href="http://bio2rdf.org/uniprot:P68403">http://bio2rdf.org/uniprot:P68403</a> Beta-II [uniprot:p68403] NAPOR [uniprot:o95319] Bruno-like protein 3 CUG-BP- and ETR-3-like factor 4	<a href="http://bio2rdf.org/uniprot:P68403">http://bio2rdf.org/uniprot:P68403</a> <a href="http://bio2rdf.org/uniprot:p68403">http://bio2rdf.org/uniprot:p68403</a> <a href="http://bio2rdf.org/uniprot:o95319">http://bio2rdf.org/uniprot:o95319</a> <a href="http://bio2rdf.org/uniprot:O95319">http://bio2rdf.org/uniprot:O95319</a> <a href="http://bio2rdf.org/uniprot:Q9BZC1">http://bio2rdf.org/uniprot:Q9BZC1</a>	Página de descripción Página de descripción Página de descripción Página de descripción Página de descripción

Tabla 4.13: Pruebas buscador semántico Falcons

## Conclusiones de Falcons

Falcons presenta algunas características destacadas que lo hacen un buscador atractivo para ciertos fines como acceso a usuarios finales, pero en el caso de acceso para desarrolladores aún es muy básico. Se puede destacar de Falcons que es un buscador bastante completo si se considera que contiene la cantidad de 7 millones de documentos semánticos en formato estandarizado RDF. Además que las entidades (clases, propiedades) son fácilmente accesibles, debido a que falcons almacena toda la información en su repositorio en forma de documentos virtuales lo que permite acceder a clases y propiedades sin requerir salir hacia un sitio externo. Sin embargo la mayoría de funcionalidades que se encuentran descritas están disponibles por medio de la página del buscador más no mediante su API. La API está orientada exclusivamente a devolver resultados a las consultas por lo que no se puede acceder a información adicional u otras características. En el caso de extracción de ontologías mediante la página no existen maneras de descargar este esquema y en el caso del API no dispone de dicho servicio para las ontologías. En caso de utilizar la búsqueda de documentos vía API se requiere una URI, lo cual es un problema debido a que requiere encontrar el documento por similitud sin conocerlo previamente. Finalmente al realizar la prueba de extracción de documentos RDF mediante el buscador solo presentó como resultados enlaces a páginas de descripción del documento semántico, y no un enlace directo hacia donde se encuentra el recurso. La falta de inclusión de las características completas en el API y la dificultad de extracción de los documentos hace que Falcons no pueda ser usado dentro del componente de búsqueda y selección de ontologías, pese a algunas características destacadas.

## 4.4. Búsqueda dentro de un repositorio local

Las búsquedas de ontologías en repositorios externos y mediante buscadores semánticos no han logrado alcanzar buenos resultados que demuestren que pueden ser implementados como métodos de selección de ontologías dentro del *Proceso Automático de Anotación Semántica de Servicios Web* (PAASSW). Por lo que se decidió crear un repositorio local adaptado a las necesidades del presente trabajo y sobre el cual se pueda desarrollar métodos de consulta personalizados.

El planteamiento de los métodos de consulta para este caso, se han enfocado a la búsqueda por texto (Full-Text Search [6]). Este enfoque se ha adoptado puesto que dentro del repositorio, el esquema o TBOX (sección 2.2.3) de las ontologías se

representa como un conjunto de palabras permitiendo que los métodos(de consulta) puedan hacer comparaciones (*Matching*) directamente sobre sus elementos. La utilización de este enfoque representa una ventaja frente a métodos de búsquedas basados en metadatos utilizados en repositorios externos o buscadores semánticos pues evita problemas como: etiquetado erróneo de las ontologías, descripciones incompletas, falta de clasificación en dominios, entre otros.

En este enfoque de consulta, los nombres de los conceptos de las ontologías son usados como *keywords* que representan a la ontología y a partir de los cuales se realiza la búsqueda. Por otro lado los nombres de los parámetros de los servicios a anotarse funcionan como términos de consulta al repositorio. La consulta al repositorio se realiza por medio de varias técnicas de búsqueda basadas en textos que se han planteado con el objetivo de encontrar aquel método que permita obtener la ontología que se relaciona en mayor medida con el servicio a anotarse.

En la presente sección por lo tanto se describe la implementación del enfoque de búsqueda de ontologías sobre repositorios locales planteado, el cual contiene dos partes principales: primero, la creación del repositorio, donde se explica el análisis, diseño e implementación del repositorio de acuerdo con los requerimientos planteados para los distintos métodos de consulta; segundo, análisis e implementación de métodos de búsqueda basada en textos, los cuales servirán para la selección automática de ontologías. Adicionalmente, se trata la creación de un Servicio Web que facilite la ejecución de las consultas sobre el repositorio.

#### 4.4.1. Creación de un repositorio local

Para realizar la búsqueda dentro un repositorio local, primero es necesario comenzar por la creación de dicho repositorio. En esta sección se tratan los procedimientos necesarios para elaborar un repositorio propio de ontologías, que cumpla con los requerimientos establecidos en el presente trabajo.

##### 4.4.1.1. Fuente de las ontologías a utilizar

Como se mencionó en la sección 4.2 la plataforma CKAN posee uno de los principales repositorios de *datasets* libres en Internet, que se pueden acceder mediante Datahub. Muchos de los *datasets* presentes son ontologías que además pueden ser accedidas mediante un SPARQL Endpoint, requisito necesario para usarse dentro del PAASSW.

Para obtener las ontologías que contienen los SPARQL Endpoints se requirió la ayuda de SPARQL Endpoints Status<sup>35</sup>, proyecto que se encarga del monitoreo de los Endpoints registrados en CKAN y el cual contiene la lista de la mayor parte de los SPARQL Endpoints públicos disponibles en Internet. Esta lista de ontologías se obtuvo mediante una consulta al API REST<sup>36</sup> del proyecto mencionado, el cual proporciona una respuesta en formato JavaScript Object Notation (JSON) de todos los SPARQL Endpoints registrados en Datahub (CKAN). Una muestra de la respuesta de este Servicio Web se puede visualizar en el Segmento de código 4.4.

```

1   ...
2 {
3     "uri": "http://dblp.rkbexplorer.com/sparql/" ,
4     "datasets": [
5       { "uri": "http://thedatahub.org/dataset/rkb-explorer-dblp" ,
6         "label": "DBLP Computer Science Bibliography (RKBExplorer)" ,
7       } ] },
8   { "uri": "http://dbpedia.org/sparql" ,
9     "datasets": [
10       { "uri": "http://thedatahub.org/dataset/dbpedia" ,
11         "label": "DBpedia" } ] },
12   { "uri": "http://dbsnp.bio2rdf.org/sparql" ,
13     "datasets": [
14       { "uri": "http://thedatahub.org/dataset/bio2rdf-dbsnp" ,
15         "label": "Bio2RDF::Dbsnp" } ]
16   },
17   ...

```

Segmento de Código 4.4: Respuesta de la API de SPARQL Endpoints Status

#### 4.4.1.2. Diseño del repositorio

El diseño interno del repositorio debe dar soporte al enfoque de búsqueda planteado (Búsqueda basada en textos), es decir debe almacenar el TBOX de las ontologías sobre modelos que permitan la ejecución de métodos de consulta basadas en texto. Varios de dichos métodos se consideraron en el presente trabajo, y se detallan en la sección 4.4.2. Considerando los requerimientos de información de estos métodos se crearon dos modelos para representar a las ontologías, los cuales se explican a continuación:

<sup>35</sup><http://sparqles.ai.wu.ac.at>

<sup>36</sup><http://sparqles.ai.wu.ac.at/api/endpoint/list>

**Esquema simplificado.** Es un subconjunto del esquema de las ontologías que considera solamente los dos elementos principales del TBOX (sección 2.2.3) de las ontologías: clases y propiedades, ignorando las relaciones existentes entre estos, es decir, se consideran a las clases y propiedades como elementos independientes que no están relacionados. El modelo ER utilizado para almacenar esta información de las ontologías se muestra en la figura 4.3, este modelo tiene tres tablas: *Ontology* que representa a las ontologías almacenadas en el repositorio, *Class* que contiene a las clases de las ontologías y *Property* que contiene a las propiedades de la ontología. Este modelo permite mantener separados los conceptos (clases y propiedades) para los métodos de búsqueda que se basan en la comparación directa entre los nombres de los conceptos y los parámetros de los Servicios Web(Ver sección 4.4.2.3).

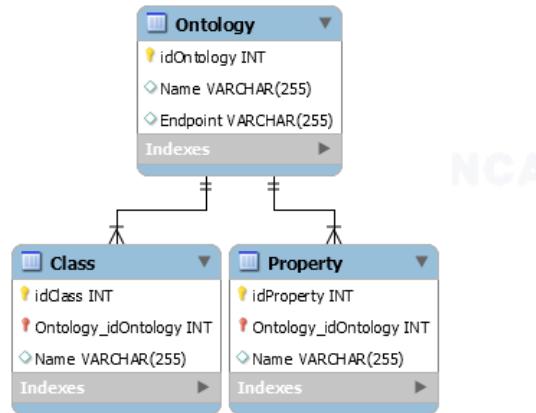


Figura 4.3: Modelo relacional: Esquema simplificado

**Texto simple.** En este modelo las ontologías son representadas como cadenas de texto simples, donde sus conceptos (clases y propiedades) son concatenadas en un único campo texto. En este modelo los conceptos se transforman en un conjunto de *keywords* sobre las cuales se ejecutarán las consultas. El modelo ER utilizado para almacenar las ontologías de muestra en la figura 4.4. Este modelo consta de una tabla única (*Ontology*), cuyas filas representarán a cada una de las ontologías del repositorio. Al representar a cada ontología como una cadena de texto (campo *Concepts*), este modelo permite la ejecución de métodos de búsqueda que utilizan indexación de textos.

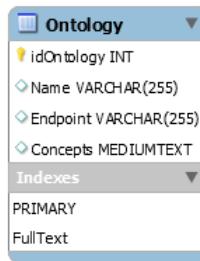


Figura 4.4: Modelo relacional: Texto Simple

Finalmente, se creó una representación alternativa del modelo de Texto simple en formato de archivo XML, con el fin de facilitar el proceso de indexación en motores de búsqueda de textos que utilicen este formato. El formato XML definido se muestra en el segmento de código 4.5. Este formato contiene los mismos campos que el modelo ER.

```

1 <doc>
2   <field name="id" value=">{id}</field>
3   <field name="name" value=">{name}</field>
4   <field name="endpoint" value=">{endpoint}</field>
5   <field name="concepts" value=">{concepts}</field>
6 </doc>
```

Segmento de Código 4.5: Representación en XML del modelo: Texto Simple

#### 4.4.1.3. Extracción del esquema (TBOX) de las ontologías

A continuación se define el procedimiento a seguir para la obtención del esquema de las ontologías, es decir, cómo consultar los SPARQL Endpoints de las ontologías para obtener los conceptos que se almacenan en el repositorio. Para la extracción del esquema o TBOX de las ontologías se usó como base el estándar RDF y RDFS<sup>37</sup>, debido a que este estándar es utilizado para la creación de ontologías en el contexto de las tecnologías de la Web Semántica [84]. La extracción del esquema de las ontologías tiene dos partes: la extracción de clases y la extracción de propiedades.

**Extracción de clases:** Para la obtención de las clases de las ontologías se utilizó la propiedad de RDFS: *rdf:type*, la cual es usada para definir relaciones de tipo “es un”, es decir, permite definir las instancias de una clase dentro de la ontología. Por ejemplo en la figura 4.5 se muestra la creación de dos instancias

<sup>37</sup>RDF Schema: Expansión de RDF para la descripción de vocabularios

de la clase *Molecular Function*: *ATP Binding* y *Electron carrier activity*, la cual se representa en RDF Turtle<sup>38</sup> en el segmento de código 4.6.

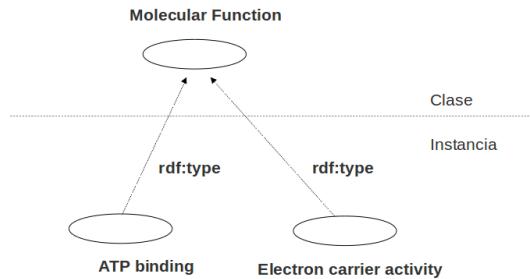


Figura 4.5: Ejemplo de instanciación de clases con RDF

```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 <ATP binding> rdf:type <Molecular Function>.
3 <Electron carrier activity> rdf:type <Molecular Function>.

```

Segmento de Código 4.6: Representación en Turtle de la creación de instancias

Para la obtención de las clases de las ontologías mediante esta propiedad se utilizó la consulta SPARQL mostrada en el segmento de código 4.7. Nótese que el estándar SPARQL establece una representación alternativa para *rdf:type* con la utilización de la partícula 'a'. Esta consulta obtiene una lista con todos los recursos (variable ?aClass), que han sido usados para generar instancias (variable auxiliar ?instance), es decir, son las clases o tipos de recursos.

```

1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 SELECT distinct ?aClass
3 WHERE {
4     ?instance a rdfs:Class
5 }

```

Segmento de Código 4.7: Extracción de Clases

**Extracción de propiedades:** La obtención de las propiedades se realizó a partir de la representación RDF. Como se mencionó en 2.2.4.1, RDF representa la información como tripletas, en las que una propiedad enlaza dos recursos. Esta forma de representación simplifica la obtención de las propiedades de una ontología, puesto que todos los recursos que se encuentren en medio de una tripla son considerados propiedades.

<sup>38</sup>Turtle: representación liviana de RDF

Para la extracción de las propiedades de las ontologías se utilizó la consulta SPARQL 4.8. Esta consulta retorna una lista con todos los recursos (variable ?aProperty) que sirven de enlace a otros recursos (?resource1 y ?resource2) dentro de las tripletas de la ontología.

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2         SELECT distinct ?aProperty
3 WHERE {
4     ?resource1 ?aProperty ?resource2
5 }
```

Segmento de Código 4.8: Extracción de Propiedades

#### 4.4.1.4. Almacenamiento de las ontologías en el repositorio

Para almacenar los conceptos extraídos de las ontologías se definieron dos modelos: Esquema simplificado y texto simple. El almacenamiento en el modelo de Esquema Simple consiste en la utilización directa del modelo ER definido, es decir, insertar los nombres de las propiedades y clases en las tablas de la figura 4.3. Para el almacenamiento en el modelo de Texto Simple, se definieron tres formas para representar las ontologías como cadenas de texto.

*Concatenación de conceptos.* En esta forma de representación, los nombres de los conceptos de las ontologías son concatenados en una cadena de texto, utilizando como separador un espacio en blanco. En la figura 4.6 se muestra un ejemplo de esta forma de representación.

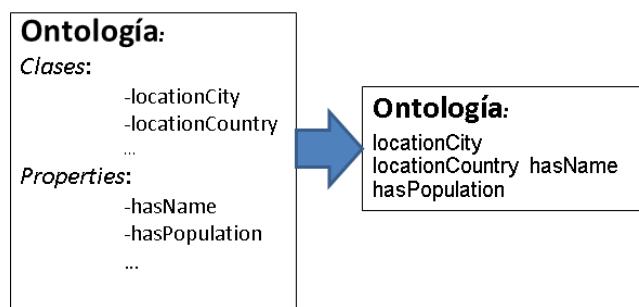


Figura 4.6: Ejemplo de representación: Concatenación de conceptos

*División de términos.* En esta representación se concatenan los nombres de los conceptos al igual que en la representación anterior, con la diferencia que cada nombre es procesado para dividirlo en palabras. Debido a que la mayoría de los nombres de conceptos están compuestos por un grupo de palabras (términos),

por ejemplo: 'hasName', 'location\_city', etc. Para la división de los términos se utilizaron los estilos de escritura: CamelCase<sup>39</sup> y UnderCase<sup>40</sup>, debido a que son los más comunes para el nombrado de elementos en la Web Semántica [52]. Continuando con el ejemplo anterior, en la figura 4.7 se muestra la aplicación de esta representación de ontologías.

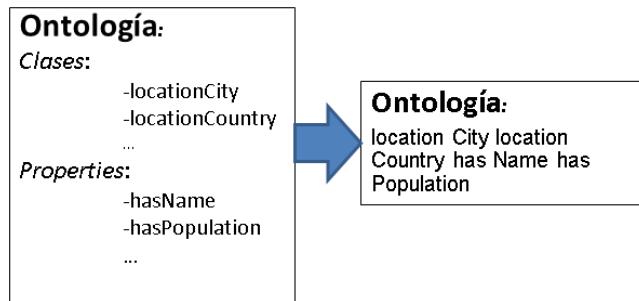


Figura 4.7: Ejemplo de representación: División de términos

*Palabras únicas.* Esta forma de representación también se basa en la división de términos, pero le agrega una etapa más de procesamiento, donde se eliminan las palabras repetidas. Dentro de los conceptos de las ontologías es común utilizar palabras repetidas para nombrar los conceptos, por ejemplo: 'locationCity', 'locationCountry', 'locationState'. Estas palabras facilitan la comprensión del significado de un concepto para los humanos, pero para en la ejecución de técnicas de búsqueda por texto (enfoque seleccionado) pueden afectar la ponderación de resultados. Es por esto que en esta representación se ejecutó un filtro de palabras repetidas, para evitar esta posible alteración en los resultados. En la figura 4.8 se muestra la aplicación de esta representación sobre la ontología utilizada como ejemplo.

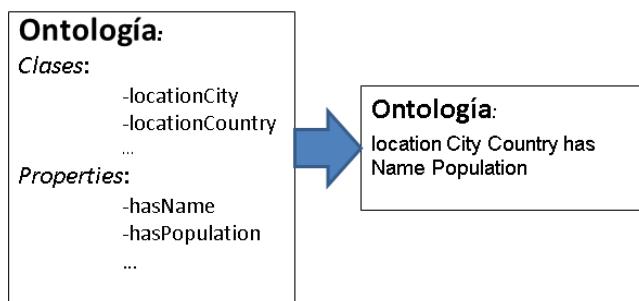


Figura 4.8: Ejemplo de representación: Palabras únicas

<sup>39</sup>Utiliza el cambios MinúsculaMayúscula para separar las palabras

<sup>40</sup>Utiliza guiones '-' o guiones bajos '\_' para separar las palabras

Para el almacenamiento de todas las formas de representación de las ontologías dentro del repositorio se utilizaron siete medios diferentes, los que se muestran en la tabla 4.14. Donde cada medio contiene una forma diferente de representar la misma ontología.

Nombre	Tipo	Modelo	Descripción
OR0	Base de datos	Esquema Simplificado	Las ontologías se almacenan directamente en el modelo ER 4.3.
OR1	Base de datos	Texto Simple	Las ontologías se almacenan en el modelo ER 4.4. Utilizando la representación en texto: Concatenación de conceptos
OR2	Base de datos	Texto Simple	Las ontologías se almacenan en el modelo ER 4.4. Utilizando la representación en texto: División de términos.
OR3	Base de datos	Texto Simple	Las ontologías se almacenan en el modelo ER 4.4. Utilizando la representación en texto: Palabras únicas.
OR1.xml	Archivo XML	Texto Simple	Las ontologías se almacenan en el modelo XML 4.5. Utilizando la representación en texto: Concatenación de conceptos
OR2.xml	Archivo XML	Texto Simple	Las ontologías se almacenan en el modelo XML 4.5. Utilizando la representación en texto: División de términos.
OR3.xml	Archivo XML	Texto Simple	Las ontologías se almacenan en el modelo XML 4.5. Utilizando la representación en texto: Palabras únicas.

Tabla 4.14: Medios de almacenamiento de las ontologías

#### 4.4.1.5. Proceso de Carga y actualización del repositorio

Para la carga de las ontologías en el repositorio se creó un programa que permite la generación de *scripts* (SQL y archivos XML), los cuales se ejecutan sobre el repositorio (Base de datos y motor de búsqueda por textos) y realizan la carga inicial y actualización de la información. En la figura 4.9 se muestra la operación de dicho programa (descarga de esquemas). Este programa realiza las siguientes actividades:

- Obtención de la lista de Ontologías a través del Servicio Web de SPARQL Endpoints Status. (Sección 4.4.1.1)
- Extracción del esquema de las ontologías, usando consultas SPARQL (Sección 4.4.1.3).
- Generación de los *scripts* (inserción/actualización) para cada uno de los siete medios planteados (Sección 4.4.1.4).

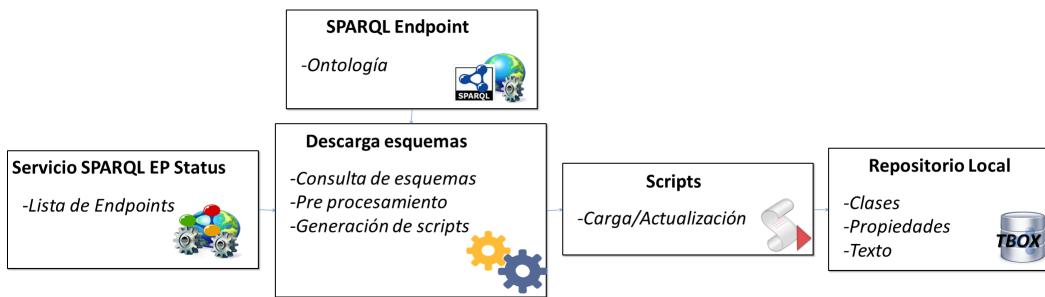


Figura 4.9: Esquema del repositorio

#### 4.4.1.6. Implementación

Para la realización del programa de generación de los *scripts* de carga/actualización se tomaron en cuenta las siguientes consideraciones:

- Se utilizó el lenguaje de programación Java debido a que la mayor parte de las librerías para el manejo de tecnologías de la Web Semántica tienen soporte para este lenguaje.
- Para la consulta de los SPARQL Endpoints se utilizó el Framework JENA<sup>41</sup>, debido a que permite la interacción con las ontologías a través del lenguaje de consulta SPARQL.
- La lista de Endpoints (Ontologías) se obtiene desde el Servicio Web de SPARQL Endpoints Status como un archivo JSON, el cual es procesado usando la librería JSONinJAVA<sup>42</sup>.
- Debido a que las consultas a los Endpoints tienen tiempos de espera considerables (segundos), se decidió utilizar varios hilos (50 Concurrentes, 1 hilo por Endpoint) para mejorar el rendimiento del programa.
- Se estableció un tiempo máximo de espera para las respuestas de las consultas de 500 segundos.
- El formato de los *scripts* desarrollados es SQL para las bases de datos y XML para el motor de búsqueda por textos.

<sup>41</sup><https://jena.apache.org/>

<sup>42</sup><http://www.json.org/java/>

#### 4.4.2. Implementación de los métodos de búsquedas sobre el repositorio

Como se mencionó en el marco teórico 2.5.2 existen principalmente tres formas para implementar búsqueda por textos: Usando funcionalidades especiales de los *Database Management System* (DBMS) (MySQL Full-Text), implementación manual (emparejamiento simple, medidas de similitud, *clustering*), usando motores de búsqueda especializados (Solr). Por esta razón para la implementación de la búsqueda o selección de ontologías dentro del repositorio se utilizaron estas tres formas.

El desarrollo de cada forma de búsqueda por textos requiere de distintas fuentes de información. Las búsquedas basadas en los DBMS y los motores de búsqueda de textos utilizan campos de texto para la ejecución de las búsquedas. Es por esto que estas formas de búsqueda han sido implementadas solamente con información del modelo de Texto Simple (en sus tres variantes). Por otro lado, la implementación manual de técnicas de búsqueda puede requerir bien sea de una representación estructurada de la información (modelo Esquema Simplificado) o como texto (modelo Texto Simple), dependiendo del tipo de técnica. Para el desarrollo específico de este trabajo durante la implementación manual se consideró solamente la información proveniente del modelo Esquema simplificado. En la figura 4.10 se muestra de forma sintetizada como las distintas formas de búsqueda a ser implementadas utilizan las fuentes de información (modelos, con sus variantes) planteadas en el presente trabajo.

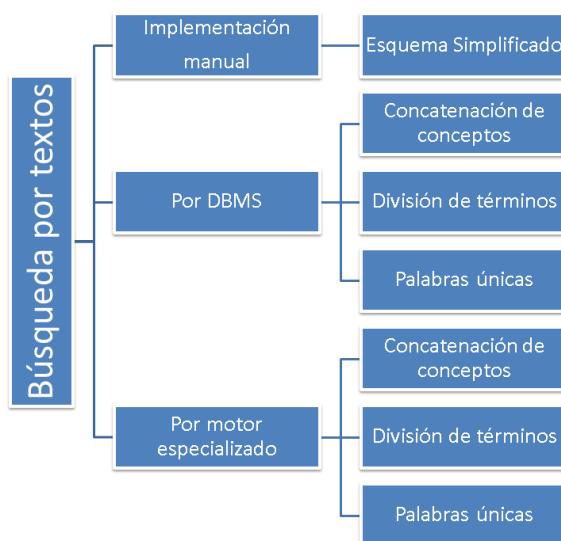


Figura 4.10: Métodos de búsqueda con sus fuentes de información

#### 4.4.2.1. Selección de ontologías a través del DBMS

Para la implementación de este tipo de búsqueda se utilizó la base de datos MySQL. Debido a que es la más extendida y usada dentro de las bases de datos relacionales de código abierto<sup>43</sup>. Esta base de datos implementa la búsqueda de textos a través de un tipo de índices especial llamado *FullText*, el cual permite tres tipos de consulta sobre los contenidos indexados (Sección 2.5.3): *Natural Language*, *Boolean Mode* y *Query Expansion*. Por esta razón, se definieron nueve consultas posibles a ser usadas sobre el repositorio, donde se consideran las tres formas de ejecución de consultas de MySQL FullText y las tres fuentes de información definidas (modelo Texto Simple en sus tres variantes). La figura 4.11 muestra un esquema de las consultas planteadas con el modo de búsqueda y la fuente de información.

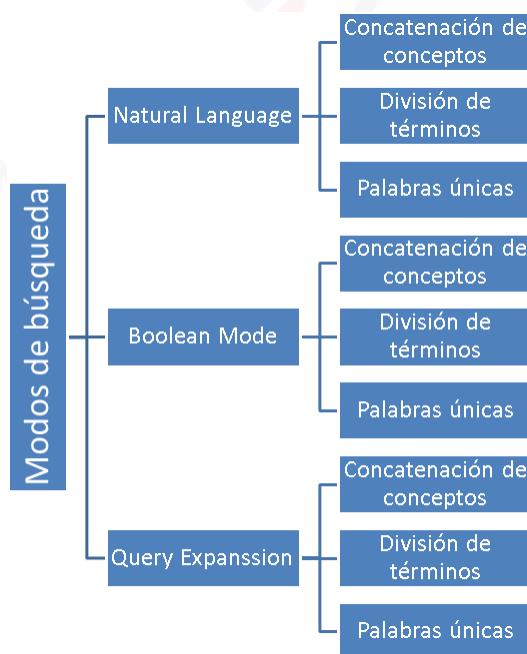


Figura 4.11: Consultas sobre MySQL

La ejecución de búsquedas usando MySQL FullText requiere de dos operadores que deben ser agregados en las consultas SQL a la base de datos: *match(...)* *against('...')*. El operador *match* indica el campo de la tabla sobre el cual se realizará la búsqueda, el cual debe ser de tipo texto y tener definido un índice tipo: *FullText*. El operador *against* define los términos de búsqueda, en este se ingresa el texto a ser buscado.

<sup>43</sup><http://db-engines.com/en/ranking>

A continuación se explican las consultas implementadas para la búsqueda de ontologías utilizando MySQL FullText en cada uno de sus modos de búsqueda. Para la creación de estas consultas se utilizó como ejemplo el Servicio Web Eventful(4.1).

#### **Consulta utilizando Natural Language**

*Natural Language* es el modo de búsqueda predeterminado de MySQL, es por esto que no requiere de otras palabras reservadas para poder ser usado. Las consultas creadas con este modo de búsqueda sobre las variantes del modelo Texto Simple se muestran en los segmentos de código: Concatenación de conceptos A.1, División de términos A.2, Palabras únicas A.3

#### **Consulta utilizando Boolean Mode**

Para definir las búsquedas como *Boolean Mode* es necesario incluir las palabras reservadas *in boolean mode* después del texto a ser buscado. Las consultas creadas con este modo de búsqueda sobre las variantes del modelo Texto Simple se muestran en los segmentos de código: Concatenación de conceptos A.4, División de términos A.5, Palabras únicas A.6

#### **Consulta utilizando Query Expansion**

Para definir las búsquedas como *Query Expansion* es necesario incluir las palabras reservadas *with query expansion* después del texto a ser buscado. Las consultas creadas con este modo de búsqueda sobre las variantes del modelo Texto Simple se muestran en los segmentos de código: Concatenación de conceptos A.7, División de términos A.8, Palabras únicas A.9

El resultado de todas estas consultas es un listado de los ontologías (SPARQL Endpoints) más relevantes encontradas utilizando MySQL FullText dentro de las bases de datos del repositorio.

#### **4.4.2.2. Selección de ontologías mediante un motor de búsqueda de textos**

Para la implementación de esta forma de búsqueda se utilizó Solr, puesto que es la plataforma de búsqueda por textos más usada en la actualidad dentro de la comunidad open source<sup>44</sup>. Además Solr contiene una capa de Servicios Web REST, la cual facilita las actividades de inserción, actualización y consulta a este motor de búsqueda.

---

<sup>44</sup><https://zooie.wordpress.com/2009/07/06/a-comparison-of-open-source-search-engines-and-indexing-twitter/>

Para la implementación de la búsqueda de ontologías sobre Solr se debe definir a cada ontología como un Documento Solr. Es decir, cada ontología se representará como un documento cuyos campos van a ser indexados, y a través de los cuales se buscarán las ontologías. Para esto Solr utiliza una definición de esquema de documentos, el cual contiene la estructura a ser utilizada para todos los documentos. Dicho esquema establece: campos del documento, tipos de datos de cada campo, filtros a ser aplicados, *tokenizers*, etc (Sección 2.5.4.1). Para la representación de ontologías en este esquema se utilizaron los campos del modelo de Texto Simple (Segmento de código 4.5). En lo que refiere a otras configuraciones, Solr tiene una lista extensa de filtros, analizadores y *tokenizers* disponibles para diversas necesidades de búsqueda<sup>45</sup>. De dicha lista se han seleccionado un conjunto de filtros que ayudarán a mejorar los resultados de las búsquedas y que se describen en la tabla 4.15. Estos filtros se seleccionaron tras un análisis de la información que se almacena (ontologías). El archivo de configuración de esquema utilizado en este trabajo se muestra en el segmento de código B.1 dentro del Apéndice B.

Adicionalmente se cambió la configuración de Solr para que el operador de consulta predeterminado sea “OR”, esto con el fin de evitar que las consultas no retornen resultados debido a la presencia de ciertos términos desconocidos dentro del repositorio. El uso de este operador implica que la aparición de todos los términos o palabras buscadas dentro de las ontologías no sea obligatorio.

Finalmente, una configuración adicional que se requiere realizar sobre Solr es el número de palabras máximo que se pueden consultar. Solr por defecto solamente permite 1024 términos de búsqueda por consulta. Sin embargo, la lista de parámetros enriquecidos de los Servicios Web pueden sobrepasar este límite es por esto que se redefinió el límite a 4096 palabras.

Tipo	Nombre	Descripción
Tokenizer	WhitespaceTokenizerFactory	Divide el texto en palabras según los espacios en blanco.
Filtro	StopFilterFactory	Filtrar las palabras vacías del texto ( <i>stopwords</i> ).
Filtro	WordDelimiterFilterFactory	Divide las palabras del texto en sub palabras usando las convenciones: CamelCase y Under Score.
Filtro	LowerCaseFilterFactory	Transforma el texto a minúsculas (Búsqueda <i>Case Insensitive</i> ).
Filtro	PorterStemFilterFactory	Normalización de palabras usando el Algoritmo de Porter.

Tabla 4.15: Filtros usados en Solr

---

<sup>45</sup><https://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters>

### Consulta al motor de búsqueda

Para la consulta del motor de búsqueda se utilizó el Servicio Web REST de consulta que provee Solr. Este servicio permite enviar una consulta POST al servidor y recibir la lista de documentos (ontologías) en formato JSON ordenados por relevancia. La consulta consiste en un conjunto de palabras (términos de búsqueda) enviadas dentro de la petición POST, en la cual también se deberá especificar el nombre de la colección a consultar, esto debido a que se generaron tres colecciones dentro del Solr, una por cada tipo de representación de las ontologías (Ver representaciones en XML 4.14).

Una consulta (Petición POST) se puede ver en las tabla 4.16 donde se utiliza como ejemplo al Servicio Web Eventful. Esta misma consulta puede realizarse también en una petición GET como se muestra en el segmento del código 4.9. El ejemplo planteado consulta a la colección 1 (concatenación de conceptos), consultas similares se realizan para las colecciones 2 (división de términos) y 3 (palabras únicas).

Propiedad	Valor
URL	http://localhost:8983/solr/collection1/select
Wt	Json
Indent	True
Q	concepts: app key location placement locating .....

Tabla 4.16: Petición POST a la primera colección: Concatenación de conceptos

```
1 http://localhost:8983/solr/collection1/select?wt=json&indent=true&q=
  concepts:+app+key+location+placement+locating .....
```

Segmento de Código 4.9: Petición GET a la primera colección: Concatenación de conceptos

#### 4.4.2.3. Selección de ontologías a través de técnicas de *matching* personalizadas

La implementación manual de técnicas de búsqueda de ontologías a diferencia de las alternativas de búsqueda mediante el uso de herramientas (DBMS, Motores de Búsqueda) requieren de la implementación personalizada de técnicas de emparejamiento (*matching*). Las técnicas de *matching* son procesos que permiten conocer el nivel de similitud que existen entre dos conjuntos de elementos, son

la base de las técnicas de búsqueda pues permiten determinar en función de los elementos existentes, cual es el de mayor relevancia con respecto al conjunto de elementos buscado para así brindar un resultado. Por esta razón, con el fin de explorar diferentes opciones de implementación manual mediante la variación de técnicas de emparejamiento se han implementado 3 alternativas: emparejamiento simple (igualdad), emparejamiento por similitud y clustering.

### Emparejamiento Simple

Un primer acercamiento para determinar la similitud existente entre los elementos del servicio y las ontologías por ser fácil de concebir e implementar es el emparejamiento simple. El emparejamiento simple consiste básicamente en comparar cada uno de los elementos en este caso de las ontologías con los elementos del servicios de forma directa. Mediante la comparación de sus elementos se puede saber exactamente cuál es el número de palabras que coinciden dentro del esquema de la ontología, con los elementos del servicio y de esta manera seleccionar aquella ontología que tenga mayores coincidencias.

La implementación de la comparación simple se llevó a cabo en el lenguaje JAVA con ayuda del comparador de texto *equals*. La función *equals* retorna un valor booleano dependiendo si las dos cadenas que se comparan (ontologías y servicios) son iguales o no. Los elementos que se ingresan a este método usados durante el proceso de matching corresponden a los atributos del servicio luego de haber pasado el proceso de descripción sintáctica y enriquecimiento, por otro lado los elementos que representan a las ontologías que están formados por las propiedades y entidades de las mismas, son extraídos de la base de datos mediante consulta al esquema simplificado (Ver sección 4.4.1.2 ). Al final de todas las comparaciones entre elementos del servicio y la ontología, se suman el número de coincidencias exitosas obtenidas durante el proceso de comparación y se asigna este resultado como puntaje para la ontología.

Entre las ventajas de este método se encuentra principalmente la facilidad de implementación, además de ser exacto pues compara solo palabras que coinciden en su totalidad.

Entre las desventajas de este método destaca que es ineficiente pues debe realizar una gran cantidad de comparaciones dependiendo del número de elementos. Formalmente posee una complejidad cuadrática lo que significa que aumenta considerablemente el tiempo de ejecución mientras mayor sea el número de elementos pertenecientes a las ontologías o servicios. Adicionalmente se debe tener

en cuenta, que no considera pequeñas variaciones en las palabras que puede ser causado por errores de ingreso pero que representan el mismo significado.

### Emparejamiento con medidas de similitud

En algunos casos al comparar dos conjuntos de palabras como se hizo anteriormente, se pudo notar que aunque las palabras comparadas, no llegaron a coincidir exactamente entre sí, podían llegar a representar la misma palabra o una muy similar. Esto puede llegar a suceder por la mala escritura de algunas palabras, abreviaciones o pequeñas variaciones dentro de las palabras, que al final llevan a desechar la igualdad. Es por esta razón, que se ha planteado otro método de comparación, en el que se pretende usar métricas de similitud para validar cuan cercana o parecida es una palabra de otra y de esta manera mejorar el margen de similitud entre las palabras.

Para utilizar las métricas de similitud se ha utilizado la librería abierta Simmetrics<sup>46</sup>, la cual posee algunos algoritmos que miden la similitud entre dos cadenas de caracteres y devuelven una salida entre 0.0 y 1.0 según su similitud. Los algoritmos que se utilizan de la librería son:

- Jaro
- Jaro-Winkler
- Levenshtein Distance
- Monge Elkan
- Needleman-Wunch
- Smith-Waterman-Gotoh

Una breve descripción de las medidas de similitud se puede encontrar en la parte del marco teórico (tabla 2.2). Para generar una medida que determine la similitud de una palabra con otra se utilizó el conjunto de medidas descritas anteriormente mediante el cálculo de su promedio. Se ha realizado la elección de usar todos los métodos mencionados como métrica de similitud, debido a que el rendimiento de los algoritmos está muy ligado al tipo del problema de texto que desea resolver y ningún algoritmo es mejor que los demás en todos los casos [2]. Esta estrategia fue tomada de los experimentos realizados en [9], donde se realiza un experimento con la métricas de similitud de forma conjunta.

---

<sup>46</sup><http://sourceforge.net/projects/simmetrics/>

### ***Clustering***

Una forma de conocer la semejanza entre documentos y por consiguiente sobre conjuntos de palabras como se explica en 2.5.2.2, es la aplicación de técnicas de *clustering* [53]. Las técnicas de *clustering* consisten en formar grupos (*clusters*) en función de las características que poseen los datos, agrupando los elementos dentro de un mismo grupo o clase mientras mayor sea el número de características que comparten y separándolos de otros con menor similitud.

En este planteamiento la idea de usar el método de *clustering* tiene como objetivo agrupar los elementos obtenidos del servicio, con los elementos de la ontología que comparte el mayor número de características en común. De esta manera se podría conocer la ontología que más se adapta al servicio que se pretende anotar en base al *cluster* en el que se encuentra. Para la aplicación del *cluster* se utilizó la librería de Weka<sup>47</sup>, la cual posee potentes herramientas de *clustering*.

### **Procedimiento de *Clustering***

Para el proceso de *clustering* se optó por utilizar la librería de Weka sobre IDE de java (NETBEANS) para el desarrollo de un programa que automatice todo el proceso de *clustering*, esto es: la toma de datos, la generación de *clusters* y obtención de resultados (Ver figura 4.12). Sin embargo, el procedimiento puede realizarse siguiendo los mismos pasos en caso de hacerlo con ayuda de la interfaz o consola. A continuación se describirá cada uno de los pasos seguidos para el desarrollo de un *cluster* con Weka.

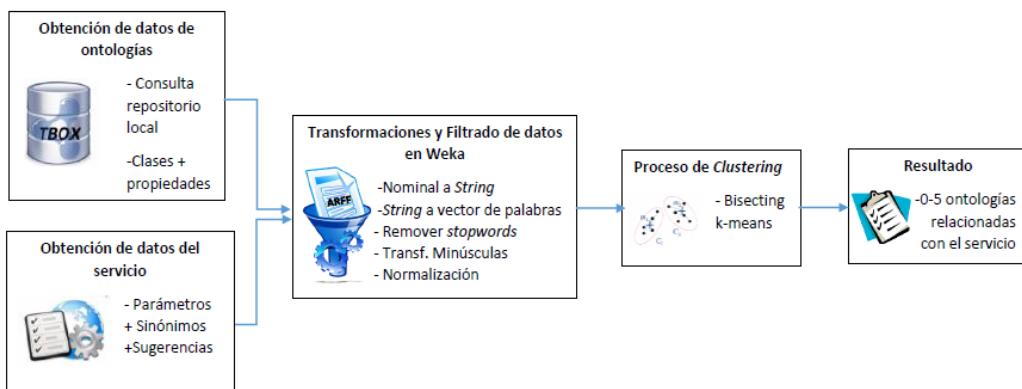


Figura 4.12: Esquema del proceso de clusterización utilizando la librería de Weka

<sup>47</sup><http://www.cs.waikato.ac.nz/ml/weka/>

### **Obtención de datos**

Los datos de las ontologías requeridos para la formación de *clusters* se encuentran en el repositorio local dentro del esquema simplificado (como se trató en 4.4.1.2) por lo que es necesario acceder a la base de datos para extraer sus elementos. Weka cuenta con comandos que permiten conectarse a las base de datos directamente, para lo cual es necesario la configuración del archivo de conexión y la adición de la librería adecuada.

Una vez tomada la información de las ontologías de la base de datos, Weka crea una variable de tipo *Instance* donde almacena toda la información consultada. Los datos leídos dentro de la variable de tipo *Instance* son definidos de forma especial dentro de memoria para ser entendidos por Weka, esto es en el formato ARFF (Attribute-Relation File Format)<sup>48</sup>.

En este tipo de formato están contenidos algunos tipos de datos como numéricos, cadenas de caracteres y nominal. Debido a que Weka reconoce el tipo de datos automáticamente cuando hace lectura desde una base de datos, suele asignar a los mismos el tipo de dato nominal. Este tipo de datos no permite ser usado como entrada a algunos métodos como el de *clustering*. Por lo tanto, previo al ingreso al proceso de *clustering* se procedió a realizar una transformación adicional de tipos de datos de Nominal a cadena de caracteres (*string*), para lo cual se utilizó un filtro llamado *NominalToString*.

Luego de haber realizado la transformación, se procedió a añadir los datos del servicio en el mismo archivo juntos con los datos de ontología mediante concatenación y siguiendo el formato ARFF.

Con todos los datos disponibles y siguiendo el proceso de *clustering* mencionado en 2.5.2.2.1, se procede a aplicar una serie de filtros que permitan el correcto manejo de las palabras por el programa y el desarrollo correcto del proceso de *clustering*. Los filtros aplicados fueron los siguientes:

**Cadena de caracteres a vectores de palabras :** Uno de los filtros principales para el manejo de texto dentro de Weka es *StringtoWordVector*, que permite reconocer a las palabras de un texto como un conjunto de atributos que representan a la palabra y a sus pesos.

---

<sup>48</sup><http://www.cs.waikato.ac.nz/ml/weka/arff.html>

**Eliminación de *stopwords*:** Se aplica este filtro para eliminar palabras comunes que no aportan mucha información en el texto. En el caso de los datos tratados se pudo comprobar que hay un gran número de palabras representadas con una letra o letras y números que más bien pueden ser considerados como ruido.

**Filtro para el cambio de palabras a minúsculas :** Para el caso en que se requiere comprobar simplemente las palabras mas no la forma en que se encuentran escritas como el caso de mayúsculas y minúsculas, es aconsejable que se opte por la conversión a una de las opciones. En este caso se transformó todas las palabras a minúscula, caso contrario el proceso de *clustering* que es *keysensitive* reconocerá como palabras diferentes dependiendo de comó esté escrita.

**Normalización de las palabras** Debido a que los elementos dentro de las ontologías pueden variar en número ampliamente<sup>49</sup>, se considera conveniente utilizar la opción de normalización. Con ayuda de la normalización se pretende evitar que las ontologías con mayor número de elementos sean las que posean mayor peso sobre las demás.

### ***Generación de clusters***

Después del tratamiento que se realizaron sobre las entradas, se procede a la configuración y creación de los *clusters*. Para la creación del *cluster* se seleccionó uno de los algoritmos disponibles por Weka, el simple k-means con la métrica de distancia “Euclidea”. Al método de simple k-means convencional se le hizo una pequeña variación en su forma de ejecución basada en la propuesta de los autores Steinbach, Karypis, Kumar en [75] conocido como “Bisecting k-means”. Se seleccionó el método Bisecting k-means debido a que es bastante conocido y utilizado en el ámbito de *clustering* y gracias a su variación según la pruebas realizadas por los autores anteriormente mencionados, se puede llegar a tener un buen rendimiento, incluso mejor frente a métodos híbridos y de *clustering* mediante jerarquías.

### ***Proceso de ejecución***

Con el clúster configurado, el procedimiento que sigue el Bisecting K-means hasta obtener el resultados es el siguiente:

---

<sup>49</sup>Se pudo notar que existen ontologías con más de 100 atributos, mientras que otras poseen menos de 10 atributos.

1. Las ontologías junto con los datos del servicio ejecutan el proceso de *clustering* con el algoritmo simple k-means hasta formar  $3^{50}$  clusters.
2. Las ontologías que se encuentren dentro del clúster junto con los datos del servicio continuarán el proceso de *clustering* descrito en 1, mientras los demás clusters se descartarán. Proceso que se realizará hasta que se cumpla la condición en el paso 3.
3. Si el número de elementos dentro del cluster en el que se encuentra los datos del servicio es menor a 5, el proceso terminará, devolviendo a los elementos que conforman el cluster como resultado.

Se puede ver una representación visual del proceso en la figura 4.13. Respecto a la respuesta se optó por tomar hasta cinco ontologías, debido a que es un número manejable de clusters y lo suficientemente grande para realizar comparaciones de los resultados con las demás técnicas.

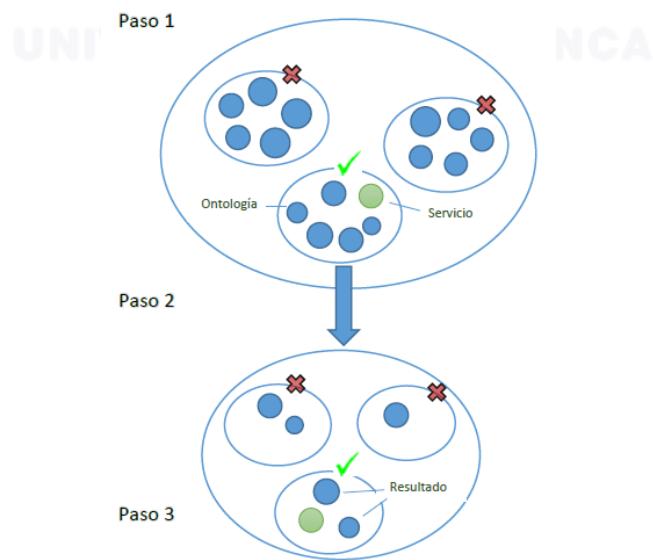


Figura 4.13: Pasos de la variación del Bisecting k-means

### **Obtención de Resultados**

Las cinco ontologías restantes o menos, luego de realizada la ejecución del proceso de *clustering* son las ontologías seleccionadas por el método como las más idóneas para su anotación. Se considera de esta manera debido a que estas

---

<sup>50</sup>El número de clusters seleccionado fue decidido en base a los tiempos de respuesta y carga del procesador. En el caso de usar dos el tiempo aumentaba notablemente.

ontologías fueron las que se conservaron durante el mayor número de iteraciones, descartando probablemente a los que compartían menos elementos en común. En este caso no es posible definir cuál es la mejor ontología de entre las encontradas, debido a que se mide la distancia hacia el centroide del *cluster* mas no entre sus elementos. Por lo tanto todos los clústeres poseerán la misma puntuación.

#### 4.4.3. Servicio Web para el acceso a los métodos de búsqueda

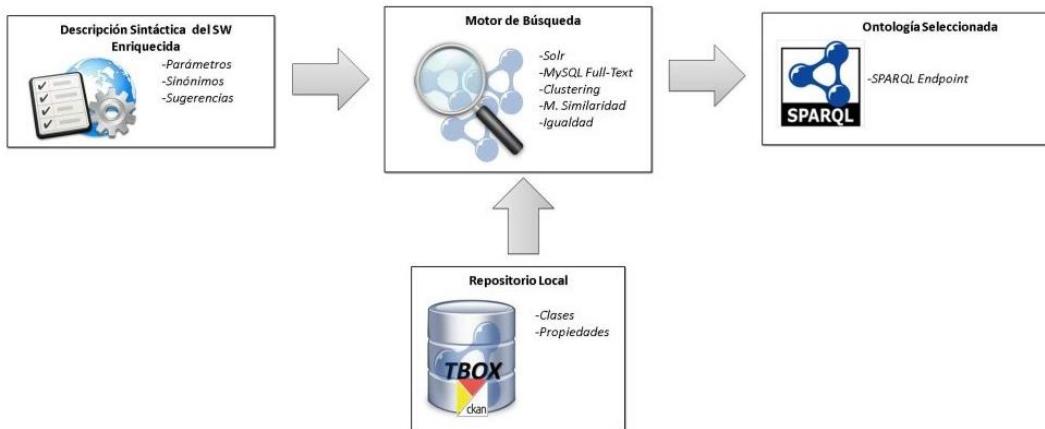


Figura 4.14: Esquema de búsqueda basada en textos

Para acceder fácilmente a la funcionalidad de todos los métodos de búsqueda y selección de ontologías tratados en este capítulo, es necesaria la creación de una interfaz común que transparente la complejidad de los métodos al usuario y que permita su acceso por medio de la plataforma. En este caso se ha implementado un Servicio Web de tipo REST con el acceso a los 15 métodos de selección de ontologías tratados, el cual ha sido acoplado a la plataforma de manera similar al resto de Servicios Web externos.

El Servicio Web recibirá como entrada la descripción sintáctica enriquecida de los Servicios Web (a anotarse) y como respuesta retornara la URL del SPARQL Endpoint activo mejor puntuado según el método que se defina al momento de consultar, (ver figura 4.14). La utilidad de crear dicho servicio se refleja en dos escenarios: primero, al ejecutar el proceso de anotación semántica, donde se encarga de seleccionar automáticamente una ontología para el Servicio Web a anotarse; segundo, cuando un usuario busque ontologías para un determinado servicio usando la interfaz gráfica de la plataforma (Ver sección 3.2.3.1).

#### 4.4.3.1. Parámetros del Servicio Web

La entrada principal para el servicio de selección de ontologías es el listado de los parámetros (enriquecidos) de un Servicio Web. Adicionalmente se definieron dos parámetros para uso interno del servicio: primero, el número del método de búsqueda, que permite escoger entre los diversos métodos de consulta desarrollados (Sección 4.4.2), los números de método definidos se muestran en la tabla 4.18; segundo, la URL del Servicio Web consultado, que se usa dentro de una caché interna de resultados (Ver sección 4.4.3.3). Los parámetros definidos para el servicio se pueden ver a más detalle en la tabla 4.17.

Parámetro	Tipo	Descripción
WSURL	Entrada, texto	URL del Servicio Web consultado
PARAMS	Entrada, texto	Lista de parámetros Enriquecidos del servicio separados por ','
METHOD	Entrada, Número	Número de método a ser usado.
OUTPUT	Salida, texto	URL del Endpoint mejor puntuado.

Tabla 4.17: Parámetros del Servicio Web

Número	Método
0	MySQL FullText Natural Language sobre Texto Simple
1	MySQL FullText Natural Language sobre Texto con división de términos
2	MySQL FullText Natural Language sobre Texto con eliminación de palabras repetidas
3	MySQL FullText Query Expansion sobre Texto Simple
4	MySQL FullText Query Expansion sobre Texto con división de términos
5	MySQL FullText Query Expansion sobre Texto con eliminación de palabras repetidas
6	MySQL FullText Boolean Mode sobre Texto Simple
7	MySQL FullText Boolean Mode sobre Texto con división de términos
8	MySQL FullText Boolean Mode sobre Texto con eliminación de palabras repetidas
9	Solr sobre Texto Simple
10	Solr sobre Texto con división de términos
11	Solr sobre Texto con eliminación de palabras repetidas
12	Clustering
13	Emparejamiento con medidas de similitud
14	Emparejamiento simple
-1	Método recomendado: Solr sobre Texto Simple (Ver sección 5.2.3).

Tabla 4.18: Tabla de métodos de búsqueda implementados con sus números

#### 4.4.3.2. Diseño interno

Con el fin de estandarizar el desarrollo del acceso a cada uno de los métodos de búsqueda implementados dentro del servicio, se decidió usar patrones de diseño de software[22]. En este caso, todos los métodos de búsqueda planteados deben tener una interfaz de acceso genérica, y a la vez cada uno debe poseer una implementación específica de su comportamiento, por esto se utilizó el patrón de diseño *Strategy*. Este patrón de software que permite definir una interfaz común para un conjunto de objetos y que según las peticiones que reciba adopta el comportamiento de uno de dichos objetos dinámicamente. El diseño creado usando un diagrama de clases UML se puede observar en la figura 4.15.

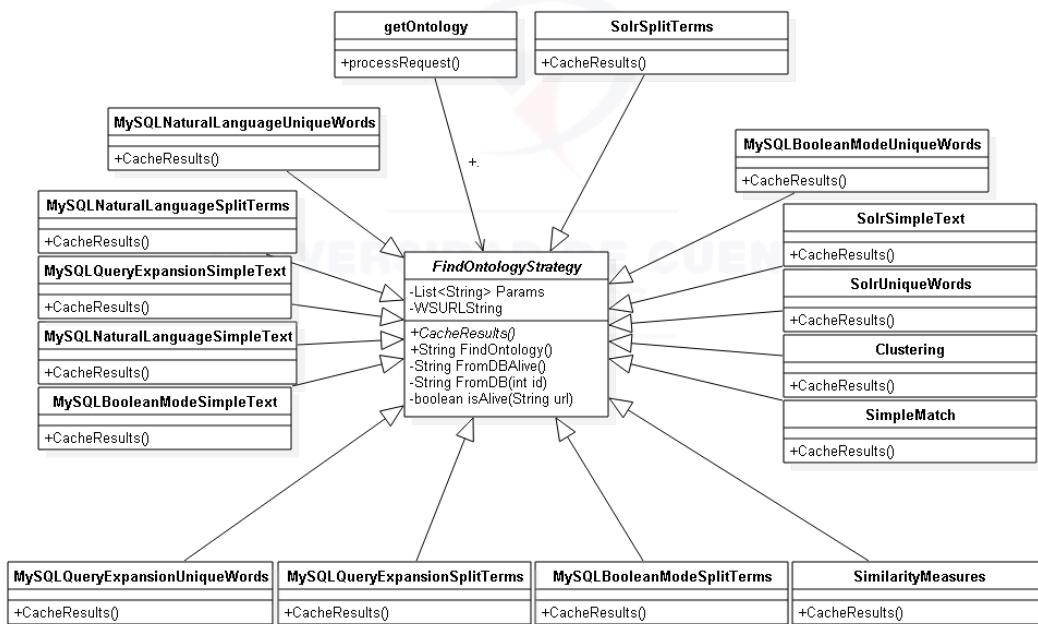


Figura 4.15: Diseño interno del Servicio Web

#### 4.4.3.3. Tabla para almacenamiento temporal de resultados (Caché)

La excesiva cantidad de tiempo que requieren los métodos de búsqueda basados en emparejamiento (simple y con medidas de similitud) hace necesario la creación de una tabla de almacenamiento temporal de resultados. Los métodos de emparejamiento tanto por similitud como por emparejamiento (sección 4.4.2.3) requieren la comparación término a término contra todas las ontologías, esto implica que los tiempos de ejecución estén en el orden de minutos o inclusive horas. Con el objetivo de mejorar el rendimiento se decidió crear una tabla de resultados.

En esta tabla se almacena y consulta los resultados de las consultas previamente ejecutadas. De esta manera los métodos de búsqueda solo se ejecutarán una sola vez y para posteriores consultas se utiliza el contenido de la tabla de resultados. El modelo Entidad Relación de la tabla de resultados se muestra en la figura 4.16.

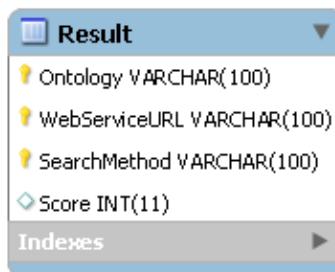


Figura 4.16: Tabla de resultados (Caché)

### Consideraciones de implementación

Para la creación del Servicio Web se tomaron en cuenta los siguiente aspectos:

- Al igual que en el programa de descarga de esquemas (Sección 4.4.1.5) se utilizó el lenguaje de programación Java debido a que cuenta con librerías que facilitan la ejecución de las consultas como: Conector a la base de datos, Cliente HTTP y soporte para tecnologías semánticas.
- Para la creación del Servicio Web se utilizó la tecnología JSP<sup>51</sup>.
- Para facilitar el desarrollo se utilizó la herramienta Maven y el IDE de desarrollo NetBeans.
- Para la conexión con la base de datos MySQL se utilizó JDBC<sup>52</sup> que permite ejecutar operaciones de consulta e inserción.
- Para la conexión con la API REST del motor de búsqueda Solr se utilizó el cliente HTTP Apache<sup>53</sup> y para procesar el resultado se utilizó la librería JSONinJAVA.
- Se verificó que los SPARQL Endpoints retornados estén activos. Para esto a los Endpoints retornados por los métodos de búsqueda se les aplica

<sup>51</sup><http://www.oracle.com/technetwork/java/javaee/jsp/>

<sup>52</sup><http://dev.mysql.com/downloads/connector/j/>

<sup>53</sup><https://hc.apache.org/>

una consulta de prueba (Usando JENA) para determinar si se encuentran activos.

- Se configuró el servicio para que funcione con peticiones POST debido a los límites de las peticiones GET en cuanto al tamaño de las consultas.

## 4.5. Resumen de los métodos tratados para la búsqueda y selección de ontologías

Para finalizar el capítulo 4, se presenta un resumen de los distintos enfoques que se han analizado durante la búsqueda de un método para la selección automática de ontologías, el cual se encuentra contenido en la tabla 4.19 y donde se presenta cada una de las alternativas tratadas juntos con sus variaciones, las ventajas y desventajas que se han encontrado durante su implementación.

Como conclusión del análisis de los distintos enfoques se puede decir que la implementación de métodos de búsqueda de ontologías sobre un repositorio local es la más viable, pues brinda la flexibilidad necesaria para adaptar las diferentes propuestas y aproximarlas a los resultados requeridos. Difiriendo de esta manera de las demás propuestas orientadas a reutilizar elementos desarrollados que al ser más rígidas, dificultan su adopción.

Métodos de búsqueda de ontologías	Alternativas planteadas	Ventajas	Problemas
Búsqueda en repositorios de ontologías externos	<ul style="list-style-type: none"> <li>■ Búsqueda directa</li> <li>■ Busq. Dominio</li> <li>■ Busq. Palabras Clave</li> <li>■ Busq. Conceptos</li> <li>■ Busq. Extracción de Entidades</li> </ul>	<ul style="list-style-type: none"> <li>■ Uso de alternativas ya implementadas.</li> <li>■ Aprovechamiento de un buscador embebido disponible para la extracción de información semántica</li> <li>■ Consideración de metadatos para la búsqueda.</li> </ul>	<ul style="list-style-type: none"> <li>■ Clasificación o ámbito de ontologías no definida estrictamente.</li> <li>■ Problemas conforme crece el número de entrada de datos para la búsqueda.</li> <li>■ Resultados no presentan en todos los casos los requerimientos necesarios (SPARQL Endpoint).</li> <li>■ Difícil personalización y ajuste en las herramientas para obtener los resultados deseados. (Alchemy, GATE)</li> <li>■ Información de metadatos incompleta.</li> </ul>
Búsqueda de ontologías mediante buscadores semánticos	<ul style="list-style-type: none"> <li>■ Watson</li> <li>■ FalconS</li> </ul>	<ul style="list-style-type: none"> <li>■ Solución ya implementada y probada en la búsqueda de información semántica.</li> <li>■ Acceso para desarrolladores vía API</li> <li>■ Disponibilidad de metadatos.</li> </ul>	<ul style="list-style-type: none"> <li>■ Funcionalidades no disponibles por falta de soporte y continuidad de la herramienta.</li> <li>■ Enlaces perdidos o desactualizados</li> <li>■ Documentos semánticos de poca extensión</li> <li>■ Limitado soporte para búsqueda sobre un gran conjunto de entradas</li> <li>■ No se puede tener acceso a SPARQL Endpoint</li> </ul>
Búsqueda mediante la implementación personalizada de repositorio local	<ul style="list-style-type: none"> <li>■ Emparejamiento Simple</li> <li>■ Emp. con Medidas de Similitud</li> <li>■ Clustering</li> <li>■ Busq. por motores de búsqueda de texto. (Solr)</li> <li>■ Busq. por DBMS. (MySQL FullText)</li> </ul>	<ul style="list-style-type: none"> <li>■ Personalización y adaptación de los métodos según las necesidades.</li> <li>■ Experimentación sin restricciones.</li> <li>■ Soporte para la búsqueda sobre un gran número de entradas.</li> <li>■ Control de datos y de requisitos mínimos para las ontologías (SPARQL Endpoint, etc.)</li> </ul>	<ul style="list-style-type: none"> <li>■ Tarea de implementación extensa.</li> <li>■ Consumo de tiempo elevado.</li> <li>■ Propenso a errores</li> <li>■ Tiempo de ejecución elevados y poco optimizados.</li> </ul>

Tabla 4.19: Resumen de los métodos para la búsqueda y selección de ontologías

## Capítulo 5

---

### Evaluación y Resultados

---

En esta sección se describe el procedimiento de evaluación realizado sobre los métodos de selección de ontologías y el proceso global de anotación semántica de Servicios Web.

#### 5.1. Métricas de evaluación

Para la evaluación del componente de selección de ontologías, se consideró los métodos de búsqueda que lo componen como sistemas clasificadores. Dichos métodos buscan seleccionar a la ontología que mejor representa a un Servicio Web de entre un conjunto de ontologías dado, por esta razón para evaluar este componente se utilizó la métrica de comparación F-measure (Valor-F, F-Score, Medida-F), valor bastante utilizado en la evaluación de sistemas de clasificación y recuperación de información. Los valores necesarios para la aplicación del F-measure se han obtenido con el apoyo de una matriz de confusión multiclasa que se describe en la sección 5.1.1 .

La métrica del F-measure se define como la media armónica entre la precisión y la exhaustividad [85], representado por la siguiente fórmula para el caso de un F-measure balanceado<sup>1</sup>.

$$F - measure = \frac{2 * (precisión * exahustividad)}{(precisión + exahustividad)}$$

Las métricas de precisión y exhaustividad son las métricas básicas para medir el nivel de resultados obtenidos con respecto a los esperados, y se definen como:

---

<sup>1</sup>El F-measure balanceado hace referencia a cuando se da igual relevancia al valor de la precisión y la exahustividad, en el cálculo de la media armónica.

**Precisión:** Proporción de documentos relevantes recuperados respecto a todos los que fueron recuperados. La precisión indica la capacidad que tiene el clasificador para reconocer los elementos que pertenecen a dicha clasificación de los que no.

$$Precisión = \frac{Documentos\_relevantes\_retornados}{Todos\_los\_documentos\_retornados}$$

Por ejemplo si se tiene dos Servicios Web de música y uno de películas, que se han clasificado con respecto a la ontología de música. La precisión en este caso sería:

Precisión para la clasificación de la ontología de música : 2/3

El valor de la precisión en este caso es de 0.66 pues 2 de los 3 servicios que recuperó eran relevantes para esa clase, mientras que 1 de los 3 no lo era.

**Exahustividad:** Indice de documentos relevantes recuperados respecto a todos los elementos relevantes que debieron ser seleccionados. La exahustividad permite conocer la capacidad del clasificador para encontrar los elementos que se ajustan a las condiciones establecidas como correctas para una determinada clase.

$$Exahustividad = \frac{Documentos\_relevantes\_retornados}{Todos\_los\_documentos\_relevantes}$$

Si tomamos el ejemplo anterior para obtener la exhaustividad con la ontología de música y sus servicios tenemos:

Exhaustividad para clasificación de la ontología de música : 2/2

El valor que se obtiene para la exahustividad es de 1 pues todos los elementos (Servicios Web) que debieron ser seleccionados para la ontología de música fueron clasificados en esa clase. En el caso de que solo hubiera obtenido como resultado un servicio de música de los dos disponibles y un servicio de películas la exhaustividad en este caso hubiera sido de 1/2.

Tomando los valores obtenidos de los ejemplos anteriormente planteados, el calculo del F-measure da como resultado :

$$\frac{2 * (2/3 * 1)}{(2/3 + 1)} = 0,79$$

El valor para el F-measure en este caso es de 0.79 que es el resultado de la media armónica entre la exhaustividad y la precisión. El valor máximo para la F-measure es de 1 que se da cuando la exhaustividad y la precisión son máximos, lo que significa que el sistema de clasificación es ideal.

La utilización de la métrica del F-measure sobre el clasificador de ontologías tiene como objetivo el descubrimiento del método con los mejores resultados en la tarea de clasificación, condición que se verá reflejada por cual obtenga un valor mayor de F-measure (promedio). El promedio de esta métrica es requerido debido a que los métodos realizan clasificación multiclasificación lo que significa que puede evaluarse el desempeño de la clasificación para cada clase correspondiente a las ontologías disponibles. En el ejemplo anterior solo se obtuvo el F-measure de la clase música, restando por calcularse esta misma métrica para las demás clases.

### 5.1.1. Matriz de confusión multiclasificación

Las matrices de confusión son la base para la representación, agrupación e identificación de resultados proporcionados por un sistema clasificador. En este se puede identificar el número de los elementos que han sido clasificados correctamente, los que han sido clasificados incorrectamente, falsos positivos y falsos negativos como se puede observar en la tabla 5.1:

Clase de Datos	Clasificado como Positivo	Clasificado como Negativo
positivo	Verdaderos Positivos	Falsos negativos
negativo	Falsos positivos	Verdaderos negativos

Tabla 5.1: Matriz de confusión

Las fórmulas de precisión y exahustividad tratadas anteriormente para matrices de confusión se pueden expresar de la siguiente manera:

$$\text{Precisión} = \frac{\text{Verdaderos\_positivos}}{\text{Verdaderos\_positivos} + \text{Falsos\_positivos}}$$

$$\text{Exahustividad} = \frac{\text{Verdaderos\_positivos}}{\text{Verdaderos\_positivos} + \text{Falsos\_Negativos}}$$

Para el caso de evaluaciones sobre sistemas multiclasificación, donde existen varias clases (ontologías en este caso) dentro de las cuales se pueden clasificar los elementos disponibles (Servicios Web), requieren de una tabla más compleja como se muestra en la tabla 5.2. Dicha tabla está organizada por dos elementos principales: las filas que representan los elementos de predicción correctos para la clasificación (Predicciones) y las columnas que reflejan los resultados realmente obtenidos por el sistema (Actuales).

		Actual				
		$M_{11}$	$M_{12}$	$M_{13}$	...	$M_{1i}$
		$M_{21}$	$M_{22}$	$M_{23}$	...	$M_{2i}$
		$M_{31}$	$M_{32}$	$M_{33}$	...	$M_{3i}$
		...	...	...	...	...
		$M_{j1}$	$M_{j2}$	$M_{j3}$	...	$M_{ji}$

Tabla 5.2: Matriz de confusión multiclasificación

En las matrices de confusión multiclasificación al igual que en las matrices de confusión simples, se puede identificar el estado de los elementos clasificados mediante la realización de sumatorias, donde :

Los verdaderos positivos se definen como  $\sum_i M_{ii}$

Los falsos negativos para cada clase  $i$  se define como  $\sum_j M_{ji}$ , donde  $j \neq i$

Los falsos positivos para cada clase  $i$  se define como  $\sum_j M_{ij}$ , donde  $j \neq i$

Como se puede apreciar en las fórmulas anteriormente descritas, los valores que se pueden obtener tanto para falsos positivos y falsos negativos son por clases. Lo que significa que se podría aplicar las fórmulas definidas anteriormente para matrices de confusión simples, con el pequeño inconveniente que solo nos daría valores parciales por cada clase. Sin embargo, dado que se requiere conocer el valor de precisión, exhaustividad y F-measure para todo el sistema de clasificación, se puede aplicar el promedio para cada uno, definidos con las siguientes fórmulas:

$$\text{Precisión\_media} = \frac{\sum_i \frac{M_{ii}}{\sum_j M_{ji}}}{i}$$

$$\text{Exhaustividad\_media} = \frac{\sum_i \frac{M_{ii}}{\sum_j M_{ij}}}{i}$$

Finalmente con la precisión media y la exhaustividad media se puede obtener el F-measure promedio [73], que es la medida necesaria para evaluar los métodos:

$$F - measure promedio = \frac{2 * (Precisión\_media * Exahustividad\_media)}{Precisión\_media + Exahustividad\_media}$$

Un ejemplo de la aplicación de matriz de confusión multiclas para los resultados de un sistema de clasificación se puede encontrar en la tabla 5.3. En este ejemplo que se basa en los resultados obtenidos en la evaluación del método de emparejamiento simple, se puede observar como son contabilizados las clasificaciones de los Servicios Web para distintos dominios de ontologías. La cantidad de elementos correctos en este caso se pueden identificar fácilmente debido a que pertenecen a los valores dentro de las celdas de color gris. Los demás elementos de la matriz de confusión que han sido calculados y resumidos se presentan en la tabla 5.4, para lo cual se han utilizado las fórmulas previamente descritas.

		Actual				
Predicción	Ontologías	Peliculas	Música	Publicaciones	Geográfico	
	Peliculas	1	3	0	0	
	Música	1	3	0	0	
	Publicaciones	1	2	0	0	
	Geográfico	0	2	0	2	

Tabla 5.3: Ejemplo de matriz de confusión multiclas

Clase	Verdaderos Positivos	Falsos Positivos	Falsos Negativos	Precisión	Exahust.
Peliculas	1	2	3	0.333	0.25
Música	3	7	1	0.3	0.75
Publicaciones	0	0	3	0	0
Geográfico	2	0	2	1	0.5
Promedio				0.408	0.375

Tabla 5.4: Resultados de las fórmulas aplicadas sobre la matriz de confusión de ejemplo

Considerando los valores de precisión media y exhaustividad media de la tabla 5.4, el valor de F-measure para el ejemplo planteado sería.

$$F - measure promedio = \frac{2 * (0,408 * 0,375)}{0,408 + 0,375} = 0,3909$$

## 5.2. Evaluación

En la presente sección se describe el procedimiento de evaluación llevado a cabo sobre los métodos de selección de ontologías tratados en la sección 4.4.2 para evaluar el rendimiento de cada método en la clasificación de Servicios Web. Posteriormente a la ejecución del proceso de evaluación descrito, se presentan los resultados obtenidos junto con su respectiva interpretación, con la cual se ha determinado el método mas idóneo para pertenecer al componente de selección automática de Servicios Web. Para finalizar esta sección adicionalmente se presenta los resultados de la ejecución del proceso de anotación semántica completo, con el objetivo de presentar el desempeño global de la propuesta sobre un conjunto de Servicios Web definidos.

### 5.2.1. Proceso de evaluación

El proceso de evaluación planteado en la presente sección tiene como objetivo valorar el rendimiento de los clasificadores de servicios descrito en el capítulo 4. Estos clasificadores tienen como objetivo la selección de la mejor ontología o la ontología más próxima con referencia a un Servicio Web. En este caso, para seleccionar una ontología se requiere realizar un proceso de clasificación previo de los servicios para determinar el mejor resultado, la evaluación queda determinada como una evaluación de sistemas clasificadores multi clase.

Como entradas a los métodos se han seleccionado 15 servicios de diferentes ámbitos, los cuales han sido obtenidos de ProgrammableWeb.com<sup>2</sup> (ver Tabla A.1 Anexos). Además se han seleccionado 5 ontologías: 4 con dominios específicos (diferentes) y 1 multidominio tomadas de DataHub<sup>3</sup>. Los dominios tanto de los servicios como de las ontologías que se han seleccionado pertenecen a los ámbitos de tipo geográfico, música, películas y publicaciones. Se determinó realizar la evaluación con este tipo de dominios debido principalmente a su disponibilidad en el repositorio y la cantidad de elementos que pertenecen a dichos dominios. En el caso del número de servicios se han considerado que la utilización de 15 servicios son suficientes puesto que con un número mayor de servicios el proceso de creación del Gold Standard se volvería muy extenso, además de que no se tiene indicios de que con un número mayor de servicios los resultados tiendan a cambiar drásticamente.

---

<sup>2</sup><http://www.programmableweb.com/>

<sup>3</sup><http://datahub.io/>

Para la valoración de cada uno de los métodos de clasificación en el proceso de evaluación, se ha tomado como métrica de referencia el F-measure promedio para cada método. Por lo tanto se ha seleccionado esta métrica como referencia puesto que el F-measure como se menciona en la sección 5.1 provee una medida general que incluye la precisión y exhaustividad del método, por lo que puede ser utilizado para determinar cuál de los métodos propuestos presenta mejores resultados. Dado que la medida del F-measure tiende a comparar los valores obtenidos con valores de referencia que representen los valores ideales para la clasificación, se ha requerido de la elaboración de un Gold Standar. El Gold Standar que se tomará como referencia ha sido realizado por un grupo de expertos, los cuales han determinado para cada servicio la ontología con la que se obtiene el mayor número de elementos en común (anotaciones). Los resultados del Gold Standar se puede ver en el anexo A.2.

### 5.2.2. Ejecución de la evaluación

Para la evaluación de los quince métodos de selección de ontologías desarrollados (ver tabla 4.18), se utilizaron dos enfoques con el fin de contribuir a la completitud de la evaluación: En el primer enfoque se consideró únicamente las ontologías de dominio específico, ignorando la ontología multidominio del conjunto, con lo cual se determinó la efectividad de cada método de selección sobre un conjunto de ontologías con dominios únicos. En el segundo enfoque se incluyó una ontología multi-dominio (EventMedia), con el afán de verificar la efectividad de los métodos desarrollados para elegir la ontología adecuada de entre un conjunto ontologías tanto multi-dominio como de dominio específico. Estas dos formas de evaluar los métodos de selección desarrollados permiten analizar de manera controlada el comportamiento de los métodos desarrollados en escenarios reales (Repositorios de ontologías en Internet).

Los resultados obtenidos de los métodos de selección desarrollados para la lista de Servicios Web planteada en esta evaluación se puede ver a detalle en una hoja de cálculo del CD anexo a este documento. En este archivo se pueden encontrar los resultados presentados en una tabla que contiene las dos primeras recomendaciones de los métodos de búsqueda para cada uno de los servicios. Esta tabla se utilizó en conjunto con la tabla de Gold Standard para la calificación de cada método con las medidas anteriormente descritas.

### 5.2.2.1. Ontologías exclusivamente de dominio específico

Para la evaluación de los 15 métodos de selección de ontologías sobre ontologías de dominio específico se crearon matrices de confusión multiclasa respectivas para cada uno de los métodos tal como se explica en la sección 5.1.1. El F-measure de cada uno de los métodos que se obtuvo con la ayuda de las matrices de confusión generadas, se presentan en la figura 5.1, en la cual se puede ver con ayuda de un gráfico de barras como varía el valor de F-measure dependiendo de los métodos evaluados. Las tablas que contienen los valores de precisión y exhaustividad detallados para cada método evaluado se pueden encontrar en el anexo A.3.

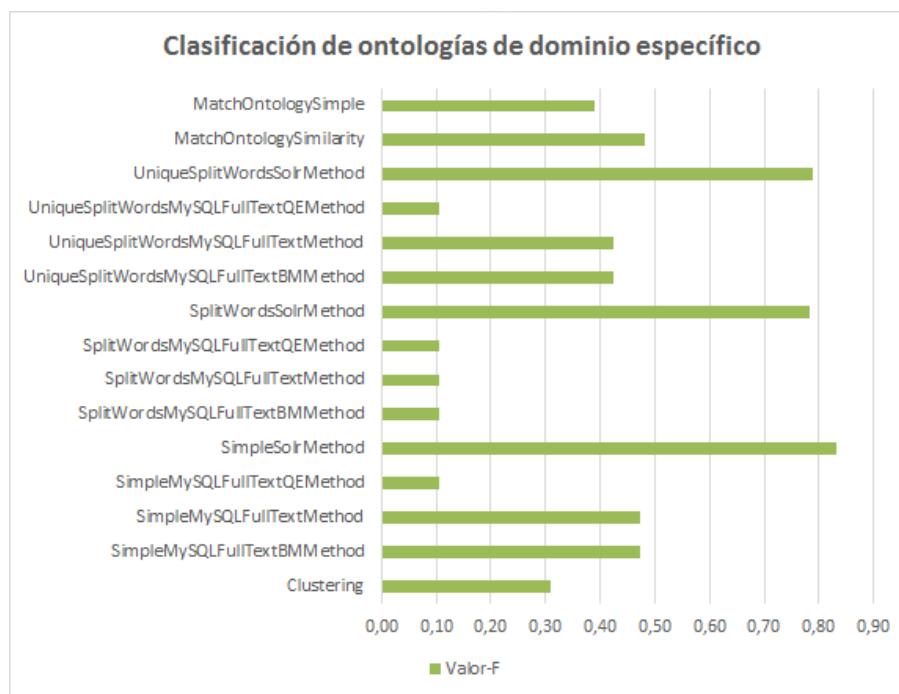


Figura 5.1: Ontologías de dominio específico: Representación en barras de los resultados.

### 5.2.2.2. Ontologías de dominio específico y multidominio

Al igual que en el enfoque anterior se evaluaron los 15 métodos de selección de ontologías, registrando sus resultados mediante matrices de confusión multiclasa. La diferencia con el enfoque anterior es que dentro de esta evaluación se usó cinco ontologías contando con una ontología multidomino EventMedia, para lo cual también se requirió la utilización de otro Gold Standar que incluya dicha ontología. Los valores de F-measure resultantes de la evaluación de modo similar

al enfoque anterior se presentan en la figura 5.2 mediante un gráfico de barras. Una tabla con los valores detallados (por cada servicio) esta evaluación se puede encontrar en la tabla A.4 dentro de Anexos.

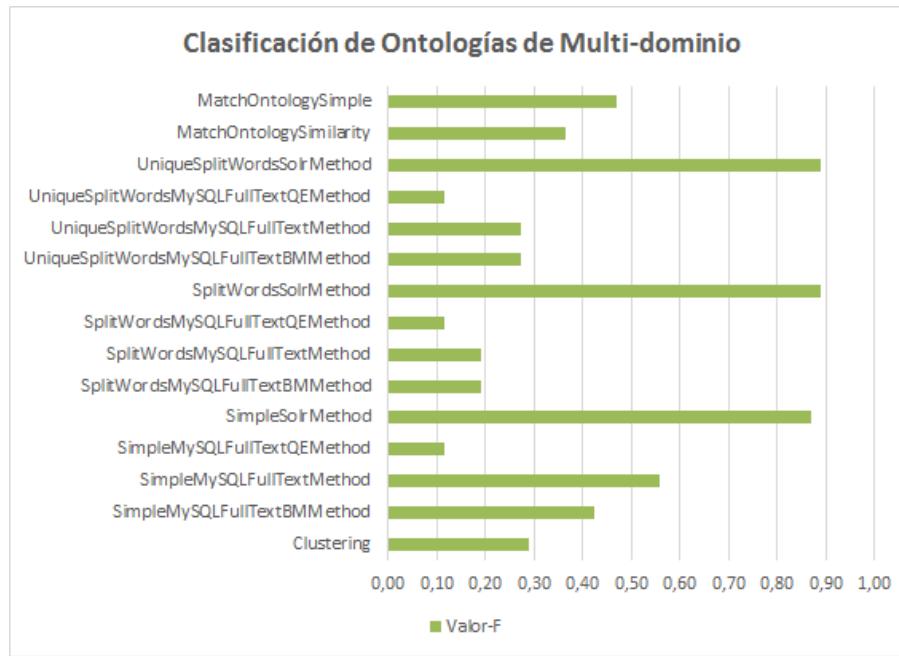


Figura 5.2: Ontologías de dominio específico y multidominio: Representación en barras de los resultados.

### 5.2.3. Interpretación de resultados

Para la evaluación de los métodos planteados en la presente tesis como se ha venido mencionado anteriormente se calculó la métrica de F-measure como medida de referencia, pues permite tener una idea general de la capacidad que tiene cada método para clasificar correctamente sus elementos (Servicios Web). Los resultados con el valor F-measure de los métodos para cada uno de los enfoque planteados se pueden observar en las figuras 5.1 y 5.2, los cuales permiten visualizar como la métrica F-measure cambia claramente según el enfoque empleado, siendo menos pronunciado entre sus variaciones.

Entrando en detalle en la información aportada por los gráficos se puede observar también, que al comparar el gráfico del primer enfoque con el del segundo enfoque no se aprecian cambios significativos. Lo que permite evidenciar que la utilización de ontologías de dominio específico y de multidominio sobre los métodos clasificadores no tienen un impacto considerable en sus resultados. Otra

de las observaciones destacadas que se pueden realizar respecto a los gráficos es que los métodos que mayor valor de F-measure han alcanzado son los basados en Solr, mientras que los que han mostrado un rendimiento menor respecto a esta métrica son los que utilizaron MySQL FullText. Para entrar en detalle respecto a los valores obtenidos para cada enfoque se ha realizado un análisis individual presentado a continuación.

#### 5.2.3.1. Métodos de búsqueda basados en Solr

Los métodos basados en el motor de búsqueda Apache Solr según muestran las graficas proveen los mejores resultados de la evaluación, tanto en el enfoque sobre ontologías de dominio específico como en la evaluación sobre ontologías con multidominio puesto que se alcanzan valores superiores a 0.8 de F-measure. Dentro de este enfoque la variación de Solr sobre texto simple se muestra ligeramente superior a los demás pues contiene el mayor valor de evaluación. Se plantea la hipótesis que este resultado se debe a que el motor de búsqueda ofrece características avanzadas como: algoritmos de Stemming, listas de stopwords, separación de términos, filtros sobre las consultas al motor, etc.

#### 5.2.3.2. Métodos de búsqueda basados en MySQL FullText

Los métodos de búsqueda basados en la funcionalidad FullText de MySQL obtuvieron mediciones de Valor F bajos (menores al 0.5). Esto se debe a que MySQL no posee funciones avanzadas de indexación para los textos, además que aplica un algoritmo de *ranking* de documentos que tiende a favorecer a los documentos más extensos.

#### 5.2.3.3. Métodos de búsqueda basados en emparejamiento simple y con medidas de similitud

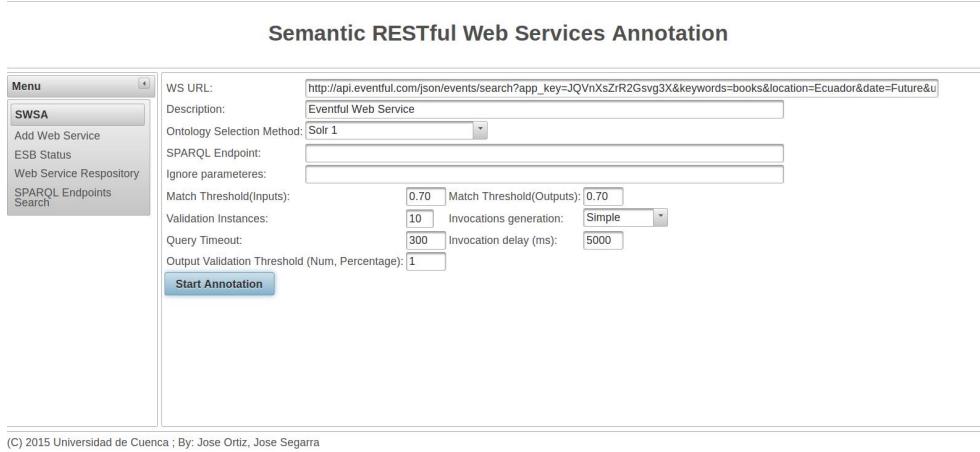
Estos métodos obtuvieron valores bajos (menores a 0.4) debido a la limitaciones en cuanto a separación de términos al momento de las comparaciones. Aunque el método de búsqueda con medidas de similitud provee cierta flexibilidad ante este problema, los resultados no son mejores con respecto al método de emparejamiento simple. Esta limitación y la ausencia de un algoritmo de *ranking* adecuado originan estos resultados.

### 5.2.3.4. Métodos de búsqueda *clustering*

Este método generó resultados bajos (menores a 0.4) debido a que al igual que los métodos de emparejamiento la ausencia de separación y filtrado de términos originó ruido en la formación de los clústeres y redujo la efectividad de este método.

### 5.2.4. Resultados del proceso de anotación semántica de Servicios Web

Una vez realizada la evaluación de la selección automática de ontologías y determinado el mejor método (Solr sobre texto simple) se ejecutó el proceso de anotación semántica completo. Como entrada al proceso se utilizó los 15 servicios disponibles junto con las 5 ontologías utilizadas anteriormente durante la evaluación. Dichos servicios fueron ingresados con ayuda de la interfaz gráfica de la plataforma y configurados como se muestra en la figura 5.3. Los resultados de las anotaciones de los servicios una vez terminado el proceso se resumen en la tabla 5.5, adicionalmente se representó el contenido de esta tabla en un gráfico de barras que se muestra en la figura 5.4.



The screenshot shows the 'Semantic RESTful Web Services Annotation' interface. On the left is a sidebar menu with options: Menu, SWSA (selected), Add Web Service, ESB Status, Web Service Repository, SPARQL Endpoints, and Search. The main panel has the following fields:

- WS URL:** http://api.eventful.com/json/events/search?app\_key=JQVnXsZrR2Gsvg3X&keywords=books&location=Ecuador&date=Future&
- Description:** Eventful Web Service
- Ontology Selection Method:** Solr 1
- SPARQL Endpoint:** (empty input field)
- Ignore parameters:** (empty input field)
- Match Threshold(Inputs):** 0.70
- Validation Instances:** 10
- Query Timeout:** 300
- Match Threshold(Outputs):** 0.70
- Invocations generation:** Simple
- Invocation delay (ms):** 5000
- Output Validation Threshold (Num, Percentage):** 1

A blue 'Start Annotation' button is at the bottom left. At the very bottom of the interface, it says '(C) 2015 Universidad de Cuenca ; By: Jose Ortiz, Jose Segarra'.

Figura 5.3: Captura de los parámetros ingresados en la plataforma

Al revisar los resultados obtenidos se puede apreciar como el método de anotación semántica automática de Servicios Web REST ha llegado a anotar en promedio un 71.57 % del total de parámetros de los servicios. Anotaciones de los cuales el 58.64 % llegaron a ser validadas por el componente de validación y por lo tanto son consideradas anotaciones correctas según el planteamiento adoptado. El servicio que mayor porcentaje de anotaciones fue Geodataservice con un

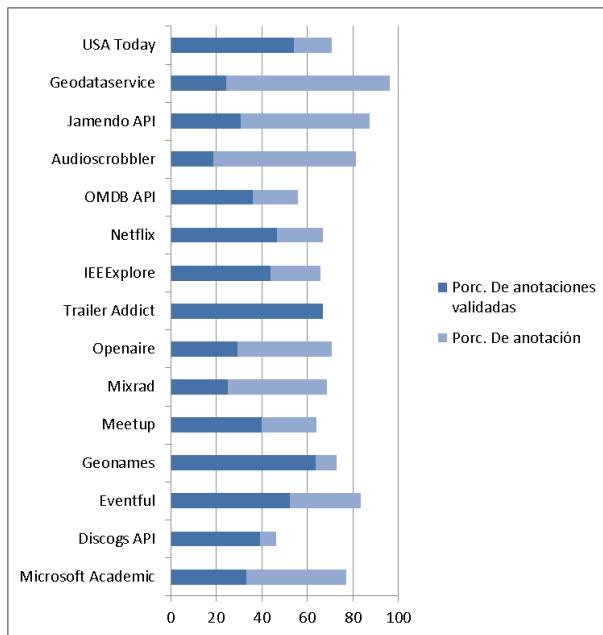


Figura 5.4: Gráfica de barras de los resultados del proceso de anotación

96.2 % de parámetros anotados, sin embargo, al validar sus anotaciones se obtuvo que sólo el 26.42 % eran válidas. Se pudo evidenciar a partir de este hecho, que un alto número de relaciones (emparejamientos) entre los parámetros del servicio y los conceptos de la ontología no implica un alto número de anotaciones válidas, por lo que es necesario un método de validación adecuado que refleje realmente las anotaciones encontradas. En el caso de las anotaciones validadas se puede encontrar que Trailer Addict es el servicio con el mayor porcentaje de anotaciones validadas con un 100 %. Tomando en consideración este alto nivel de validación y como están descritos los parámetros del servicio se ha planteado como hipótesis de que el porcentaje de validación de los servicios depende en su mayoría de como estén descritos sus parámetros. Con la utilización de parámetros completos y que contengan una descripción clara de los mismos se puede conseguir un mayor número de parámetros validados. Por otro lado, con la utilización de parámetros genéricos como “name” presentes en varios servicios, el número de anotaciones (emparejamientos) puede llegar a aumentar, pero su validación se verá reducida.

Servicios	Número de Parámetros		Param. Anotados y Validados		Param. Anot. y No Validados		Param. No Anotados		Porcent. de Anotaciones Validadas
	Entradas	Salidas	Entradas	Salidas	Entradas	Salidas	Entradas	Salidas	
Microsoft Academic	6	42	3	13	3	18	0	11	43.24
Discogs API	4	24	1	10	0	2	3	12	77.08
Eventful	5	62	4	31	1	20	0	11	46.43
GeoNames	3	19	3	11	0	2	0	6	83.58
Meetup	5	20	2	8	1	5	2	7	72.73
Mixradio	5	11	2	2	3	0	0	5	64.00
Openaire	3	14	2	3	0	7	1	4	68.75
Trailer Addict	2	7	2	4	0	0	0	3	70.59
IEEEExplore	5	27	3	11	0	7	2	9	66.67
Netflix	3	12	3	4	0	3	0	5	66.67
OMDB API	4	21	1	8	0	5	3	8	56.00
Audioscrobbler	4	12	0	3	4	6	0	3	64.29
Jamendo API	7	32	0	12	4	18	3	2	81.25
Geodatascrvice	1	183	0	45	1	131	0	7	87.18
USA Today	1	23	0	13	0	4	1	6	96.20
Totales	58	509	26	178	17	232	15	99	Prom.: 71.57
									Prom.: 58.64

Tabla 5.5: Tabla con los resultados del proceso de Anotación Semántica

## **Capítulo 6**

---

### **Conclusiones**

---

En el presente proyecto de tesis se ha llevado a cabo el planteamiento y desarrollo de un prototipo de plataforma para la anotación semántica automática de Servicios Web REST, basado en el enfoque de anotación definido en la tesis doctoral descrita en [21]. Para lo cual, inicialmente fue necesario realizar una revisión de métodos y herramientas disponibles en el estado del arte relacionado con la anotación semántica de Servicios Web, esto con el fin de encontrar y seleccionar métodos que se encaminen a dicha automatización. Posteriormente con la selección del método se ha procedido a la adaptación de la propuesta complementándola con un método de selección automática de ontologías. Para lo cual se requirió del planteamiento de varias alternativas con el fin de determinar la que aportaba mejores resultados. Finalmente, se ha llevado a cabo la implementación de la propuesta con ayuda de tecnología de vanguardia en dicho ámbito mediante un Bus de Servicios Empresarial. Al concluir con el proyecto por lo tanto se ha cumplido con la totalidad de los objetivos específicos planteados inicialmente dentro de la presente tesis, tal como se describe continuación:

- Análisis del estado del arte sobre los distintos enfoques para Anotación Semántica de Servicios Web: Se ha realizado una revisión de las diferentes propuestas en el estado del arte respecto a la anotación semántica de servicios principalmente REST y tecnologías relacionadas, encontrándose que la propuesta de Saquicela y otros provee un enfoque más orientado hacia la automatización.
- Investigación sobre distintas arquitecturas orientadas hacia servicios, patrones de integración y sobre las diferentes implementaciones requeridas para el manejo, integración y anotación de servicios: Se ha investigado diferentes

arquitecturas de tecnologías orientadas hacia el manejo de servicios, seleccionando al Bus de Servicios Empresarial Service Mix por ser la tecnología que está en vanguardia con respecto al manejo e integración de servicios y además de soportar los principales patrones de integración que se han utilizado en el proceso de implementación del prototipo.

- Implementar y extender del enfoque de Anotación Semántica de Servicios Web mediante un proceso de selección automática de ontologías: Se ha diseñado e implementado un software de anotación semántica de Servicios Web sobre un Bus de Servicios y se ha adicionado con respecto al enfoque original un componente para selección automática de ontologías.
- Prueba y evaluación del proceso de Anotación Semántica de Servicios Web con la selección automática de ontologías: Se ha evaluado la efectividad del componente de selección de ontologías a través de un Gold Standard elaborado por expertos, además se ha probado el proceso completo de anotación (con selección automática de ontologías) con un conjunto de servicios de prueba. Por lo tanto, a través de los resultados obtenidos en las dos etapas de validación del sistema se puede comprobar la validez del mismo.

A continuación se mencionan las posibles líneas futuras de trabajo que se pueden abordar en este ámbito:

- Mejorar la efectividad del componente de la validación de las anotaciones mediante la definición de nuevas reglas de validación, que estén basadas en el tipo de datos de cada parámetro, ya que la validación del prototipo propuesto se limita a comparación directa de cadenas de texto.
- Creación de una etapa de inserción de anotaciones manuales para los elementos que no han podido ser anotados automáticamente.
- Optimización en el tiempo del proceso total de anotación tomando en consideración un número mayor de servicios y ontologías de diferente dominio.
- Crear un componente para generación de descripciones semánticas y sintácticas de los Servicios Web en lenguajes formales (SAWDL, OWL-S, etc), debido a que las descripciones generadas por el prototipo se almacenan directamente en una base de datos a través de un modelo relacional.

- Incluir soporte multi lenguaje al sistema de anotación en todas sus etapas, puesto que para la implementación del prototipo únicamente se consideró el uso de ontologías y servicios en Inglés.
- Crear un componente de inferencia que permita relacionar las anotaciones resultantes de manera jerárquica usando las relaciones semánticas del modelo ontológico, puesto que el enfoque de anotación usado retorna como resultado una estructura “plana” de anotaciones sobre los parámetros.
- Relacionar la plataforma de anotación con otras herramientas que aprovechen las anotaciones semánticas generadas por ejemplo para la realización de búsqueda e invocación de servicios, así como la creación de *MashUps*.
- Con las anotaciones generadas proponer la creación de *Linked Services* y enlazar con la Web de los Datos.

## Apéndice A

---

### Anexos

---

```
1 select ONTOLOGY.ENDPOINT, match(ONTOLOGY.CONCEPTS) against ('app key  
    location placement locating .....') as score from ONTOLOGY  
order by score desc
```

Segmento de Código A.1: Consulta Natural Language sobre base datos para Concatenación de conceptos

```
1 select ONTOLOGYP.ENDPOINT, match(ONTOLOGYP.CONCEPTS) against ('app  
    key location placement locating .....') as score from ONTOLOGYP  
order by score desc
```

Segmento de Código A.2: Consulta Natural Language sobre base datos para División de términos

```
1 select ONTOLOGYPU.ENDPOINT, match(ONTOLOGYPU.CONCEPTS) against ('app  
    key location placement locating .....') as score from  
    ONTOLOGYPU order by score desc
```

Segmento de Código A.3: Consulta Natural Language sobre base datos para Palabras únicas

```
1 select ONTOLOGY.ENDPOINT, match(ONTOLOGY.CONCEPTS) against ('app key  
    location placement locating .....' in boolean mode) as score  
from ONTOLOGY order by score desc
```

Segmento de Código A.4: Consulta Boolean Mode sobre base datos para Concatenación de conceptos

```
1 select ONTOLOGYP.ENDPOINT, match(ONTOLOGYP.CONCEPTS) against ('app
key location placement locating .....' in boolean mode ) as
score from ONTOLOGYP order by score desc
```

Segmento de Código A.5: Consulta Boolean Mode sobre base datos para División de términos

```
1 select ONTOLOGYPU.ENDPOINT, match(ONTOLOGYPU.CONCEPTS) against ('app
key location placement locating .....' in boolean mode ) as
score from ONTOLOGYPU order by score desc
```

Segmento de Código A.6: Consulta Boolean Mode sobre base datos para Palabras únicas

```
1 select ONTOLOGY.ENDPOINT, match(ONTOLOGY.CONCEPTS) against ('app key
location placement locating .....' with query expansion ) as
score from ONTOLOGY order by score desc
```

Segmento de Código A.7: Consulta Query Expansion sobre base datos para Concatenación de conceptos

```
1 select ONTOLOGYP.ENDPOINT, match(ONTOLOGYP.CONCEPTS) against ('app
key location placement locating .....' with query expansion )
as score from ONTOLOGYP order by score desc
```

Segmento de Código A.8: Consulta Query Expansion sobre base datos para División de términos

```
1 select ONTOLOGYPU.ENDPOINT, match(ONTOLOGYPU.CONCEPTS) against ('app
key location placement locating .....' with query expansion )
as score from ONTOLOGYPU order by score desc
```

Segmento de Código A.9: Consulta Query Expansion sobre base datos para Palabras únicas

Servicio Web	URL de Invocación
Discogs	<a href="https://api.discogs.com/database/search?artist=Nirvana&amp;country=Canada&amp;key=uspcCNpCzMEqmkbKgbwQ&amp;secret=EIfEPC0xWUUCsunZaKoVObmGldFujgWo">https://api.discogs.com/database/ search?artist=Nirvana&amp;country=Canada&amp; &amp;key=uspcCNpCzMEqmkbKgbwQ&amp;secret= EIfEPC0xWUUCsunZaKoVObmGldFujgWo</a>
Mixradio	<a href="http://api.mixrad.io/1.x/us/recommendations/?category=artist&amp;domain=music&amp;client_id=2116baae725750b68ddb6a18297f204c&amp;album=Nevermind&amp;genre=Grunge">http://api.mixrad.io/1.x/us/recommendations/ ?category=artist&amp;domain=music&amp;client_id= 2116baae725750b68ddb6a18297f204c&amp;album= Nevermind&amp;genre=Grunge</a>
Academic Research Microsoft	<a href="http://academic.research.microsoft.com/json.svc/search?AppId=5751acf3-21f3-41e4-a244-920eb895ee08&amp;AuthorQuery=Saquicela&amp;ResultObjects=Publication&amp;PublicationContent=AllInfo&amp;StartIdx=0&amp;EndIdx=10">http://academic.research.microsoft. com/json.svc/search?AppId= 5751acf3-21f3-41e4-a244-920eb895ee08&amp; &amp;AuthorQuery=Saquicela&amp;ResultObjects= Publication&amp;PublicationContent=AllInfo&amp; &amp;StartIdx=0&amp;EndIdx=10</a>
IEEE EXPLO-RER	<a href="http://ieeexplore.ieee.org/gateway/ipsSearch.jsp?querytext=java&amp;py=2010&amp;isbn=978-1-4244-5585-0&amp;pu=IEEE&amp;ctype=Conferences">http://ieeexplore.ieee.org/gateway/ipsSearch.jsp? querytext=java&amp;py=2010&amp;isbn=978-1-4244-5585-0&amp; &amp;pu=IEEE&amp;ctype=Conferences</a>
Eventful	<a href="http://api.eventful.com/json/events/search?app_key=JQVnXsZrR2Gsvg3X&amp;keywords=books&amp;location=Ecuador&amp;date=Future&amp;units=km">http://api.eventful.com/json/events/search?app_ key=JQVnXsZrR2Gsvg3X&amp;keywords=books&amp;location= Ecuador&amp;date=Future&amp;units=km</a>
Geonames	<a href="http://api.geonames.org/searchJSON?name=ecuador&amp;country=ec&amp;username=demo">http://api.geonames.org/searchJSON?name=ecuador&amp; &amp;country=ec&amp;username=demo</a>
The Open Movie Database	<a href="http://www.omdbapi.com/?t=terminator&amp;y=2015&amp;plot=full&amp;r=json">http://www.omdbapi.com/?t=terminator&amp;y=2015&amp; &amp;plot=full&amp;r=json</a>
Netflix Roulette	<a href="http://netflixroulette.net/api/api.php?director=JoeyFigueroa&amp;actor=SylvesterStallone&amp;title=RockyV">http://netflixroulette.net/api/api.php?director= JoeyFigueroa&amp;actor=SylvesterStallone&amp;title= RockyV</a>
Audioscrobbler	<a href="http://ws.audioscrobbler.com/2.0/?method=track.getInfo&amp;api_key=8ba72c35be6900d1c968d565ad912119&amp;track=nothing&amp;artist=metallica">http://ws.audioscrobbler.com/2.0/ ?method=track.getInfo&amp;api_key= 8ba72c35be6900d1c968d565ad912119&amp;track=nothing&amp; &amp;artist=metallica</a>
Geodata	<a href="http://azure.geodataservice.net/GeoDataService.svc/GetUSDemographics?zipcode=33444">http://azure.geodataservice.net/GeoDataService. svc/GetUSDemographics?zipcode=33444</a>
USATODAY	<a href="http://api.usatoday.com/open/reviews/movies/movies/Rambo?api_key=t27s3zd4dznvjega97xk8un">http://api.usatoday.com/open/reviews/movies/ movies/Rambo?api_key=t27s3zd4dznvjega97xk8un</a>
MeetUpCities	<a href="http://api.meetup.com/2/cities?&amp;sign=true&amp;photo-host=public&amp;country=ec&amp;query=Cuenca&amp;state=azuay&amp;page=20">http://api.meetup.com/2/cities?&amp;sign=true&amp; &amp;photo-host=public&amp;country=ec&amp;query=Cuenca&amp; &amp;state=azuay&amp;page=20</a>
OpenAire	<a href="http://api.openaire.eu/search/publications?title=Java&amp;size=4&amp;model=sygma">http://api.openaire.eu/search/publications?title= Java&amp;size=4&amp;model=sygma</a>
TrailerAddict	<a href="http://api.traileraddict.com/?film=the-dark-knight&amp;count=8">http://api.traileraddict.com/?film= the-dark-knight&amp;count=8</a>
Jamendo	<a href="http://api.jamendo.com/v3.0/tracks/?client_id=4a942b9d&amp;format=jsonpretty&amp;limit=2&amp;fuzzytags=">http://api.jamendo.com/v3.0/tracks/?client_id= 4a942b9d&amp;format=jsonpretty&amp;limit=2&amp;fuzzytags=</a>
José Ortiz, José Segarra	<a href="http://www.grooveshark.com/search?query=rock&amp;speed=high+veryhigh&amp;include=musicinfo&amp;search=Nothing">groove+rock&amp;speed=high+veryhigh&amp;include= musicinfo&amp;search=Nothing</a>

Tabla A.1: Listado de Servicios Web utilizados durante la evaluación

Servicios	Ontología	Número de anotaciones
Netflix Roulette	<i>EventMedia</i>	9
	<i>GeoSparql</i>	7
	<i>LinkedBrainz</i>	4
	<i>IEEE</i>	6
	<b>LinkedMDB</b>	<b>11</b>
OPEN MOVIE DATABASE	<i>EventMedia</i>	14
	<i>GeoSparql</i>	10
	<i>LinkedBrainz</i>	10
	<i>IEEE</i>	9
	<b>LinkedMDB</b>	<b>15</b>
Geonames	<i>EventMedia</i>	9
	<i>GeoSparql</i>	8
	<i>LinkedBrainz</i>	4
	<i>IEEE</i>	2
	<b>LinkedMDB</b>	<b>6</b>
Eventful	<b>EventMedia</b>	<b>19</b>
	<i>GeoSparql</i>	10
	<i>LinkedBrainz</i>	8
	<i>IEEE</i>	9
	<b>LinkedMDB</b>	<b>3</b>
IEEE Explorer	<b>EventMedia</b>	<b>13</b>
	<i>GeoSparql</i>	8
	<i>LinkedBrainz</i>	7
	<i>IEEE</i>	12
	<b>LinkedMDB</b>	<b>2</b>
Academic Search Microsoft	<b>EventMedia</b>	<b>20</b>
	<i>GeoSparql</i>	9
	<i>LinkedBrainz</i>	10
	<i>IEEE</i>	14
	<b>LinkedMDB</b>	<b>1</b>
MixRadio	<i>EventMedia</i>	5
	<i>GeoSparql</i>	4
	<b>LinkedBrainz</b>	<b>5</b>
	<i>IEEE</i>	5
	<b>LinkedMDB</b>	<b>2</b>
Discodogs	<i>EventMedia</i>	9
	<i>GeoSparql</i>	5
	<b>LinkedBrainz</b>	<b>10</b>
	<i>IEEE</i>	7
	<b>LinkedMDB</b>	<b>5</b>
MeetUp	<i>EventMedia</i>	9
	<b>GeoSparql</b>	<b>11</b>
	<i>LinkedBrainz</i>	1
	<i>IEEE</i>	0
	<b>LinkedMDB</b>	<b>2</b>
OpenAire	<i>EventMedia</i>	5
	<i>GeoSparql</i>	2
	<i>LinkedBrainz</i>	2
	<b>IEEE</b>	<b>8</b>
	<b>LinkedMDB</b>	<b>3</b>
TrailerAddict	<i>EventMedia</i>	2
	<i>GeoSparql</i>	1
	<i>LinkedBrainz</i>	3
	<i>IEEE</i>	1
	<b>LinkedMDB</b>	<b>4</b>
Jamendo	<i>EventMedia</i>	6
	<i>GeoSparql</i>	1
	<b>LinkedBrainz</b>	<b>15</b>
	<i>IEEE</i>	3
	<b>LinkedMDB</b>	<b>3</b>
audioscrobbler	<b>EventMedia</b>	<b>9</b>
	<i>GeoSparql</i>	6
	<i>LinkedBrainz</i>	8
	<i>IEEE</i>	7
	<b>LinkedMDB</b>	<b>7</b>
USATODAY	<i>EventMedia</i>	9
	<i>GeoSparql</i>	12
	<i>LinkedBrainz</i>	9
	<i>IEEE</i>	11
	<b>LinkedMDB</b>	<b>15</b>
GeoData	<b>EventMedia</b>	<b>16</b>
	<i>GeoSparql</i>	12
	<i>LinkedBrainz</i>	3
	<i>IEEE</i>	1
	<b>LinkedMDB</b>	<b>5</b>

Tabla A.2: Gold Standard: Resultados de la anotación de los expertos

Método	Precisión	Exhaustividad	Valor-F
Clustering	0.31	0.31	0.31
SimpleMySQLFullTextBMMETHOD	0.45	0.5	0.47
SimpleMySQLFullTextMethod	0.45	0.5	0.47
SimpleMySQLFullTextQEMETHOD	0.07	0.25	0.11
SimpleSolrMethod	0.85	0.81	0.83
SplitWordsMySQLFullTextBMMETHOD	0.07	0.25	0.11
SplitWordsMySQLFullTextMethod	0.07	0.25	0.11
SplitWordsMySQLFullTextQEMETHOD	0.07	0.25	0.11
SplitWordsSolrMethod	0.82	0.75	0.78
UniqueSplitWordsMySQLFullTextBMMETHOD	0.46	0.4	0.42
UniqueSplitWordsMySQLFullTextMethod	0.46	0.4	0.42
UniqueSplitWordsMySQLFullTextQEMETHOD	0.07	0.25	0.11
UniqueSplitWordsSolrMethod	0.83	0.75	0.79
MatchOntologySimilarity	0.51	0.46	0.48
MatchOntologySimple	0.41	0.38	0.39

Tabla A.3: Resultados con ontologías de dominio específico

Método	Precisión	Exhaustividad	Valor-F
Clustering	0.27	0.31	0.29
SimpleMySQLFullTextBMMETHOD	0.4	0.46	0.42
SimpleMySQLFullTextMethod	0.55	0.57	0.56
SimpleMySQLFullTextQEMETHOD	0.08	0.2	0.11
SimpleSolrMethod	0.83	0.92	0.87
SplitWordsMySQLFullTextBMMETHOD	0.16	0.23	0.19
SplitWordsMySQLFullTextMethod	0.16	0.23	0.19
SplitWordsMySQLFullTextQEMETHOD	0.08	0.2	0.11
SplitWordsSolrMethod	0.87	0.92	0.89
UniqueSplitWordsMySQLFullTextBMMETHOD	0.3	0.25	0.27
UniqueSplitWordsMySQLFullTextMethod	0.3	0.25	0.27
UniqueSplitWordsMySQLFullTextQEMETHOD	0.08	0.2	0.11
UniqueSplitWordsSolrMethod	0.87	0.92	0.89
MatchOntologySimilarity	0.36	0.38	0.37
MatchOntologySimple	0.46	0.48	0.47

Tabla A.4: Resultados incluyendo una ontología multidominio

## Apéndice B

---

### Archivo de esquema para Solr.

---

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <schema name="example" version="1.5">
3     <field name="_version_" type="long" indexed="true" stored="true"
4     />
5     <field name="_root_" type="string" indexed="true" stored="false"
6     />
7     <field name="id" type="string" indexed="true" stored="true"
8     required="true" multiValued="false" />
9     <field name="name" type="string" indexed="false" stored="true"
10    required="true" />
11    <field name="endpoint" type="string" indexed="false" stored="true"
12    required="true" />
13    <field name="concepts" type="text_en_splitting" indexed="true"
14    stored="false" />
15    <uniqueKey>id</uniqueKey>
16    <defaultSearchField>concepts</defaultSearchField>
17    <solrQueryParser defaultOperator="OR" />
18    <fieldType name="string" class="solr.StrField" sortMissingLast="true" />
19    <fieldType name="long" class="solr.TrieLongField" precisionStep="0"
20    positionIncrementGap="0" />
21    <fieldType name="text_en_splitting" class="solr.TextField"
22    positionIncrementGap="100" autoGeneratePhraseQueries="true">
23        <analyzer type="index">
24            <tokenizer class="solrWhitespaceTokenizerFactory" />
25            <filter class="solr.StopFilterFactory"
26                ignoreCase="true"
27                words="lang/stopwords_en.txt" />
28            <filter class="solr.WordDelimiterFilterFactory"
29                generateWordParts="1" generateNumberParts="1" splitOnCaseChange="1" />
```

```
21      <filter class="solr.LowerCaseFilterFactory"/>
22      <filter class="solr.PorterStemFilterFactory"/>
23  </analyzer>
24  <analyzer type="query">
25      <tokenizer class="solrWhitespaceTokenizerFactory"/>
26      <filter class="solr.StopFilterFactory"
27          ignoreCase="true"
28          words="lang/stopwords_en.txt"/>
29      <filter class="solrWordDelimiterFilterFactory"
30          generateWordParts="1" generateNumberParts="1" splitOnCaseChange="
31          1"/>
32          <filter class="solrLowerCaseFilterFactory"/>
33          <filter class="solrPorterStemFilterFactory"/>
34      </analyzer>
    </fieldType>
</schema>
```

Segmento de Código B.1: Archivo de esquema de Solr

---

## Acrónimos

---

**B2B** *Business-to-Business.* 48, 166

**DBMS** *Database Management System.* 124, 129, 166

**EIA** *Integración de Aplicaciones Empresariales.* 48–50, 166

**ESB** *Bus de Servicios Empresarial.* 48, 50, 51, 55, 166

**GATE** *General Architecture for Text Mining.* 93, 166

**GRDDL** *Gleaning Resource Descriptions from Dialects of Languages.* 166

**hREST** *hmtl REpresentational State Transfer.* 26, 28, 30, 60, 166

**HTML** *HyperText Markup Language.* 11, 24, 26, 28, 30, 166

**HTTP** *Hypertext Transfer Protocol.* 8, 11, 24, 26–28, 50, 52, 73, 78, 86, 140, 166

**JMS** *Java Message Service.* 166

**JSON** *JavaScript Object Notation.* 24, 68–70, 108, 115, 124, 128, 166

**LOD** *Linked Open Data.* 88, 89, 94, 95, 103, 166

**MicroWSMO** *Micro Web Service Modeling Ontology.* 28, 166

**MUC** *Message Understanding Conference.* 93, 166

**NMR** *Normalized Message Router.* 52, 166

**OWL** *Ontology Web Language.* 19, 22, 23, 89, 91, 166

**PAASSW** *Proceso Automático de Anotación Semántica de Servicios Web.* 114, 115, 166

**RDF** *Resource Description Language.* 17, 19, 20, 22–24, 28, 33, 88, 89, 91, 110, 113, 117–119, 166

**REST** *REpresentational State Transfer.* 2, 7, 8, 11, 13–15, 25–28, 32, 49, 52, 108, 115, 136, 140, 166

**SA-REST** *Semantic Anotation REST.* 28, 166

**SOA** *Arquitectura Orientada a Servicios.* 48–50, 166

**SOAP** *Simple Object Access Protocol.* 2, 7–11, 13, 14, 25, 28, 49, 50, 108, 166

**SPARQL** *SPARQL Protocol and RDF Query Languaje.* 24, 90–92, 106, 107, 110, 115, 117, 118, 123, 166

**SWEET** *Semantic Web Services Editing Tool.* 2, 30, 166

**SWSAL** *Semantic Web Services Anotation Language.* 28, 166

**UDDI** *Universal Description, Discovery and Integration.* 8–10, 50, 166

**URI** *Uniform Resource Identifier.* 7, 9, 11, 20, 77, 112, 113, 166

**URL** *Uniform Resource Locator.* 11, 12, 32, 68, 72, 85, 137, 166

**USDL** *Unified Service Description Languaje.* 27, 166

**WADL** *Web Application Description Language.* 26, 28, 29, 166

**WSDL** *Web Service Description Language.* 8–10, 14, 25, 27, 32, 50, 166

**XML** *Extensive Markup Language.* 7–12, 17, 20, 24, 26–28, 32, 43, 44, 68, 108, 117, 123, 124, 129, 166

---

## Bibliografía

---

- [1] Charu C. Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In *Mining Text Data*. Springer US, 2012.
- [2] Iván Amón and Claudia Jiménez. *Funciones de Similitud sobre Cadenas de Texto: Una Comparación Basada en la Naturaleza de los Datos*. 2005.
- [3] Grigoris Antoniou and Frank Van Harmelen. *A semantic web primer*. MIT press, 2004.
- [4] Apache Lucene, [http://lucene.apache.org/core/3\\_5\\_0/api/core/org/apache/lucene/search/Similarity.html#formula\\_coord](http://lucene.apache.org/core/3_5_0/api/core/org/apache/lucene/search/Similarity.html#formula_coord). *Class Similarity*.
- [5] Alberto Los Santos Aransay. Revisión de los servicios web soap/rest: Características y rendimiento. Technical report, Universidad de Vigo, 2009.
- [6] Jeffrey Beall. The weaknesses of full-text searching. *The Journal of Academic Librarianship*, 34(5):438-444, 2008.
- [7] Djamal Benslimane, Schahram Dustdar, and Amit Sheth. Services mashups: The new generation of web applications. *IEEE Internet Computing*, 12(5):13–15, 2008.
- [8] Fernando Berzal. Clustering. Presentación para el departamento de Ciencias de la computación - Universidad de Granada.
- [9] Luis Manuel Vilches Blázquez. Metodología para la integración basada en ontologías de información de bases de datos heterogéneas en el dominio hidrográfico. Master's thesis, Escuela Técnica Superior de Ingenieros en Topografía, Geodesia y Cartografía, 2011.
- [10] Djelloul Bouchiha and Mimoun Malki. *Semantic Annotation of Web Services*. In *ICWIT 2012, Web and Information Technologies*, 2012.

- [11] Djelloul Bouchiha, Mimoun Malki, Djihad Djaa, Abdullah Alghamdi, and Khalid Alnafjan. Semantic annotation of web services: A comparative study. In Roger Lee, editor, *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, volume 492 of *Studies in Computational Intelligence*, pages 87–100. Springer International Publishing, 2013.
- [12] Dan Brickley and R.V. Guha. *RDF Schema 1.1*. Google and W3C, <http://www.w3.org/TR/rdf-schema/>.
- [13] Jorge Cardoso, A. Barros, N. May, and U. Kylau. Towards a unified service description language for the internet of services: Requirements and first developments. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 602–609, July 2010.
- [14] Dave Chappell. *Enterprise Service Bus*. O'Reilly, 2004.
- [15] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawaranay. Web services description language (wsdl) version 2.0 part 1: Core language. Technical report, W3C Recommendations, 2009.
- [16] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.
- [17] Richard Cyganiak, David Wood, and Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax*. W3C, <http://www.w3.org/TR/rdf11-concepts/>.
- [18] Abbie Barbir Daniel Austin. *Web Services Architecture Requirements*. W3C , Intel , IBM, <http://www.w3.org/TR/2002/WD-wsa-reqs-20021011>.
- [19] Mathieu d'Aquin and Natalya F. Noy. Where to publish and find ontologies? a survey of ontology libraries. *Web Semantics :Science , Services and Agents on the Wold Wide Web*, 2012.
- [20] Rafael Z. Frantz. *Integracion de aplicaciones : Un lenguaje específico de dominio para el diseño de soluciones de integración*. The Distributed Group, 2008.
- [21] Víctor Saquicela Galarza. *Semantic Annotation of RESTful and WFS OGC servicesl*. PhD thesis, Facultad de Informática - Universidad Politécnica de Madrid, 2015.

- [22] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
- [23] Jose Emilio Labra Gayo. Departamento de Informática de la Universidad de Oviedo, <http://www.di.uniovi.es/~labra/cursos/-Web20/images/VocabServiciosWeb.png>.
- [24] Alberto Fernández Gil. Descripción, descubrimiento y composición de servicios en entornos multiagente abiertos. un enfoque organizacional. Master's thesis, Departamento de Arquitectura y Tecnología de Computadores y Ciencias de la Computación e Inteligencia Artificial, 2007.
- [25] Anurag Goel. Enterprise integration eai vs. soa vs. esb.
- [26] Wael H. Gomaa and Aly A. Fahmy. Article: A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18, April 2013. Full text available.
- [27] Yuzhong Qu Gong Cheng, Weiyi Ge. *Falcons: Searching and Browsing Entities on the Semantic Web*. 2008.
- [28] Trey Grainger and Timothy Potter. *Solr in Action*. Manning Publications Co., 2014.
- [29] RDF Working Group. *Resource Description Framework (RDF)*. W3C, <http://www.w3.org/RDF/>.
- [30] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5):907–928, 1995.
- [31] Hartwig Gunzer. Introduction to web services. Technical report, Borland, 2002.
- [32] Fabio Casati Gustavo Alonso. Web services. In *Web Services - Concepts, Architectures and Applications*. Springer Verlag, 2004.
- [33] Marc Hadley. Web application description language. Technical report, Sun Microsystems Inc , W3C, 2009.

- [34] Maria Maciel Helena Lastres, José Cassiolato. *Systems of innovation for development in the knowledge era: an introduction*. In *Proceedings of the the First Globelics Academy, Ph.D. School on National Systems of Innovation and Economic Development, Lisbon, Portugal*. Georgia Institute of Technology, 2004.
- [35] Ivan Herman, Ben Adida, Manu Sporny, and Mark Birbeck. Rdfa 1.1 primer - third edition. Technical report, W3C, 2015.
- [36] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns*. Addison-Wesley Professional, 2004.
- [37] Xi-Ping Jia Hong Peng, Peng Peng. An improved measuring similarity for short text snippets and its application in clustering search engine. volume 3. Machine Learning and Cybernetics, 2008 International Conference on, 2008.
- [38] Anna Huang. Similarity measures for text document clustering. Departamento de Ciencias de la Computación - Universidad de Waikato.
- [39] Davis John and Rajasree M. S. Restdoc: Describe, discover and compose restful semantic web services using annotated documentations. *International Journal of Web & Semantic Technology (IJWesT)*, 4(1), January, 2013.
- [40] Vipul Kashyap, Christoph Bussler, and Matthew Moran. *The Semantic Web*. Springer, 2008.
- [41] Jacek Kopeck, Tomas Vitvar, and Dieter Fensel. Microwsmo:semantic description o frestful services. Technical report, WSMO Working Draft, 2008.
- [42] Jacek Kopecky, Karthik Gomadam, and Tomas Vitvar. *hRESTS: An HTML microformat for describing REST-ful web services*. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2008.
- [43] Yongxin Liao, Mario Lezoche, Herve Panetto, and Nacer Boudjlida. *Semantic Annotation Model Definition for Systems Interoperability*. In *OTM 2011 Workshops 2011 - 6th International Workshop on Enterprise Integration, Interoperability and Networking (EI2N)*, 2011.
- [44] Phillip Lord. Components of an ontology. 2010.

- [45] Maria Maleshkova, Carlos Pedrinaci, and John Domingue. Semantically annotating restful services with sweet. partially supported by the EU funding under the project SOA4All (FP7 - 215219).
- [46] Maria Maleshkova, Carlos Pedrinaci, and John Domingue. *Semantic annotation of Web APIs with SWEET*. In *6th Workshop on Scripting and Development for the Semantic Web*, 2010.
- [47] Enrico Motta Mathieu d'Aquin. *Watson, more than a Semantic Web search engine*. 2011.
- [48] Deborah L. McGuinness and Frank van Harmelen. *OWL Web Ontology Language Overview*. W3C, <http://www.w3.org/TR/2004/REC-owl-features-20040210/>.
- [49] Falko Menge. Enterprise service bus. FREE AND OPEN SOURCE SOFTWARE CONFERENCE 2007, 2007.
- [50] Senem Kumova Metin, Tarik Kisla, and Bahar Karaoglan. Named entity recognition in turkish using association measures. *Advanced Computing: An International Journal (ACIJ)*, 3(4), July, 2012.
- [51] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39-41, 1995.
- [52] Elena Montiel-Ponsoda, Daniel Vila-Suero, Boris Villazón-Terrazas, Gordon Dunsire, Elena Escolano, and Asunción Gómez-Pérez. Style guidelines for naming and labeling ontologies in the multilingual web. 2011.
- [53] Ms.K.Sruthi and Mr.B.Venkateshwar Reddy. Document clustering on various similarity measures. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2013.
- [54] MySQL, <https://dev.mysql.com/doc/internals/en/full-text-search.html>. *Full-Text Search*.
- [55] MySQL, <https://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html>. *Full-Text Search Functions*.
- [56] MySQL, <https://dev.mysql.com/doc/refman/5.6/en/innodb-fulltext-index.html>. *InnoDB FULLTEXT Indexes*.

- [57] David Nadeau. A survey of named entity recognition and classification. Elaborado para el Institute of Computational Linguistics, University of Heidelberg - Alemania.
- [58] Rafael Navarro. Rest vs web services. 2006.
- [59] Hung Nguyen. Introduction to mysql full-text search. Imagen, 2013.
- [60] Yves Lafon Nilo Mitra. *SOAP Version 1.2 Part 0: Primer (Second Edition)*. Ericsson , W3C, <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>.
- [61] Francesco Pagliaretti, Luca Spalazzi, and Gilberto Taccari. *Application of SWSAL in Semantic Annotation of RESTful Web Services*. In *In KDD Workshop on Text Mining*, 2012.
- [62] Abhijit Patil, Swapna Oundhakar, Amit Sheth, and Kunal Verma. *Meteor-s web service annotation framework*. In *Proceedings of the 13th international conference on World Wide Web*, 2004.
- [63] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet::similarity: Measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL-Demonstrations '04, pages 38–41, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.
- [64] Chris Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003.
- [65] Eric Prud'hommeaux and Andy Seaborne. *SPARQL Query Language for RDF*. W3C and Hewlett-Packard Laboratories, <http://www.w3.org/TR/rdf-sparql-query/>.
- [66] Tijs Rademakers and Jos Dirksen. *Open Source ESB in Action*. Manning Publications Co., 2008.
- [67] Victor Saquicela, Luis Manuel Vilches Blázquez, and Óscar Corcho. Adding semantic annotations into (geospatial) restful services. *Int. J. Semantic Web Inf. Syst.*, 8(2):51–71, 2012.
- [68] Hadi Khosravi Farsani Sareh Aghaei, Mohammad Ali Nematbakhsh. *EVOLUTION OF THE WORLD WIDE WEB : FROM WEB 1.0 TO WEB 4.0*. *IJWesT*, 3(1), 2012.

- [69] Amit P. Sheth, Karthik Gomadam, and Jon Lathem. Sa-rest: Semantically interoperable and easier-to-use services and mashups. *Internet Computing, IEEE*, 11(6), November, 2007.
- [70] K. Sivashanmugam, K. Verma, A. P. Sheth, and J. Miller. Adding semantics to web services standards. Kno.e.sis Publications, 2003.
- [71] Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, and John Miller. *Adding Semantics to Web Services Standards*. 2003.
- [72] Fatma Slaimi, Sana Sellami, Omar Boucelma, and Ahlem Ben Hassine. Flexible matchmaking for restful web services. On the Move to Meaningful Internet Systems: OTM 2013 Conferences, 2013.
- [73] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, 45(4):427–437, July 2009.
- [74] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. Technical report, Universidad de Minnesota, 2000.
- [75] Michael Steinbach, George Karypis, and Vipin Kumar. *A comparison of document clustering techniques*. In *5th Interop-Vlab*, 2000.
- [76] Brian Suda. Soap web services. Master's thesis, University Edinburgh - School of Informatics, 2003.
- [77] Berners-Lee T., J. Hendler, and O. Lassila. The semantic web. *Scientific american*, 284(5):28–37, 2001.
- [78] Kelvin Tan. Basic solr concepts. Tutorial escrito por un consultor de Solr, 2015.
- [79] Adolfo Lozano Tello. Ontologías en la web semántica. I Jornadas de Ingeniería Web' 01, 2001.
- [80] Jesús Cáceres Tello. La web semántica y el lenguaje rdf. 2006.
- [81] TIBCO. El papel de un bus de servicios empresariales (esb) en una soa. Technical report, TIBCO Software, 2008.

- [82] Domingo Suárez Torres. Enterprise integration patterns apache camel. Presentación utilizada por SynergyJ en la SG Conference Expo 09.
- [83] W3C, [http://www.w3.org/wiki/Search\\_engines](http://www.w3.org/wiki/Search_engines). *Search engines*.
- [84] World Wide Web Consortium (W3C), <http://www.w3.org/standards/semanticweb/ontology>. *Semantic Web Standards: Vocabularies*.
- [85] Ethan Zhang and Yi Zhang. F-measure. In LING LIU and M.TAMER ÖZSU, editors, *Encyclopedia of Database Systems*, pages 1147–1147. Springer US, 2009.

