UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD AZCAPOTZALCO

DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

Clasificación de servicios web semánticos mediante ontologías

REPORTE FINAL DE PROYECTO TERMINAL

Alumno
Erick Sánchez Estrada

Asesora
Dra. Maricela Claudia Bravo Contreras
Departamento de Sistemas

Abril 2013

Índice

1. Introducción

Un sistema de clasificación es una aplicación que organiza la información que se ingresa en clases mediante el uso de determinadas características comunes. El objetivo de la clasificación es que dicha información pueda ser más fácil de obtener sin tener que consultar toda la información para encontrar lo que se está buscando.

En el caso de una aplicación construida sobre una ontología, dicha aplicación agrupa la información en base a la ontología haciendo el proceso más automatizado ya que el usuario que proporciona el recurso o información no tiene que decidir cuál es la clasificación sino que es la misma aplicación que decide dónde clasifica dicha información.

El Lenguaje de Ontologías Web (OWL[1]) es un lenguaje de marcado que está pensado para ser usado cuando la información contenida en los documentos necesita ser procesada por las aplicaciones, a diferencia de situaciones en donde el contenido sólo necesita ser presentado a los humanos.

Los servicios web semánticos están formados por un servicio web y una anotación semántica sobre dicho servicio. La anotación semántica consiste en asociar conceptos y relaciones de una ontología con parámetros y operaciones de in servicio web.

Los servicios web semánticos surgen de la necesidad de realizar las operaciones de descubrimiento, selección, composición, negociación, invocación, monitorización y recuperación semiautomática de los servicios web tradicionales.

OWL-S[2] se basa en la definición de varias ontologías escritas en OWL que permiten la descripción de servicios web semánticos en diferentes niveles de abstracción. Según éste enfoque, la anotación semántica trata de dar respuesta a tres cuestiones esenciales para cualquier servicio web: qué ofrece el servicio, cómo funciona el servicio y cómo se interactúa con él.

La web Semántica debe permitir un mayor acceso no sólo al contenido sino también a los servicios de la Web. Los usuarios y los agentes de software debe ser capaz de descubrir, invocar, redactar y supervisar los recursos web que ofrecen servicios particulares y que tiene propiedades particulares, y debe ser capaz de hacerlo con un alto grado de automatización, si así se desea.

[1] Web Ontology Language
[2] Semantic Markup for Web Services

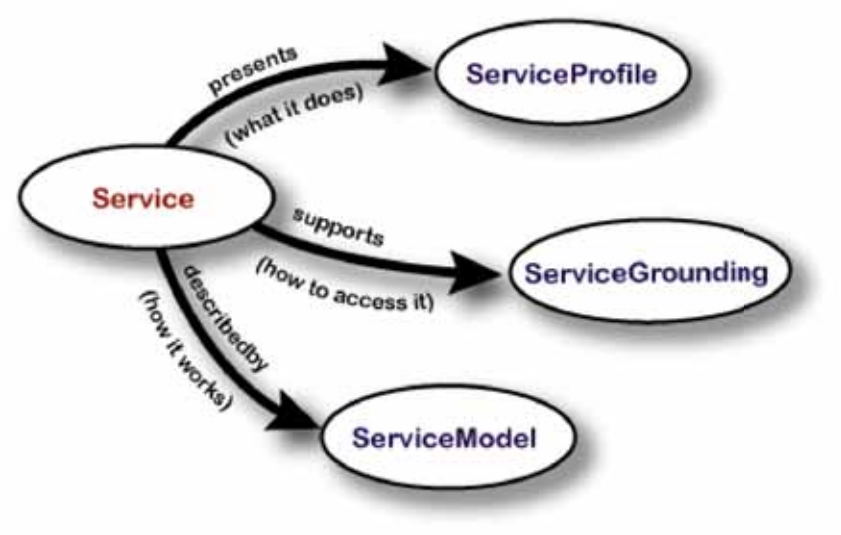

Figura 1. Estructura ontológica

Teniendo estos tres tipos de conocimiento es posible extraer una descripción de las funcionalidades que realiza el servicio OWL-S[2] y es así como se puede realizar una clasificación ya sea tomando en cuenta la descripción contenida en el *profile* o la información del *process model*; en el caso del *grounding* no se ésta tomando en cuenta como una variable para la clasificación ya que éste elemento describe la información para realizar la conexión con el servicio web y éstos datos no nos proporcionan una descripción intrínseca del servicio.

1.1. Objetivo general

Diseñar y construir un sistema de clasificación automatizada de servicios web semánticos OWL-S mediante ontologías.

1.2. Objetivos específicos

- Diseño e implementación de un repositorio local para el almacenamiento de servicios web semánticos clasificados.
- Diseñar e Implementar un módulo de clasificación basado en reglas
- Diseñar e implementar un analizador sintáctico para extraer los elementos relevantes de los servicios web semánticos.
- Diseñar e implementar una ontología que contenga la clasificación de los servicios web previamente analizados.
- Diseñar e Implementar un módulo de búsquedas de servicios web semánticos.

1.3. Trabajos relacionados

En la siguiente tabla (ver Tabla 1) se muestran los trabajos relacionados con el proyecto.

Nombre del proyecto	Similitudes	Diferencias
Extracción automatizada y representación de servicios Web mediante ontologías[2]	Este proyecto construye una ontología para representar de ésta forma servicios web para implementar a cada uno de éstos una aplicación cliente que pueda consumir dichos servicios	No clasifica estos servicios ni implementa un buscador de éstos
Solución de problemas de mediana dificultad utilizando composición de servicios web[3]	Este proyecto utiliza archivos de descripción de servicios web para la construcción de composición de servicios para la resolución de problemas	No realiza una organización de dichos servicios
Modelo Multidimensional para la representación de Perfiles de Aprendizaje y Estilos de Pensamiento mediante Ontologías y Reglas de Inferencia[4]	Este proyecto trabaja con reglas de inferencia para construir una base de conocimiento para identificar los perfiles de aprendizaje de un estudiante	No realiza clasificación ni organización

Tabla 1. Trabajos relacionados

2. Desarrollo

2.1. Diseño

2.1.1. Arquitectura del sistema

La arquitectura del sistema se muestra en la Figura 2, en donde la base del sistema está soportada por la tecnología Java SE[7] y la capa de la vista donde se utilizó la API Swing.

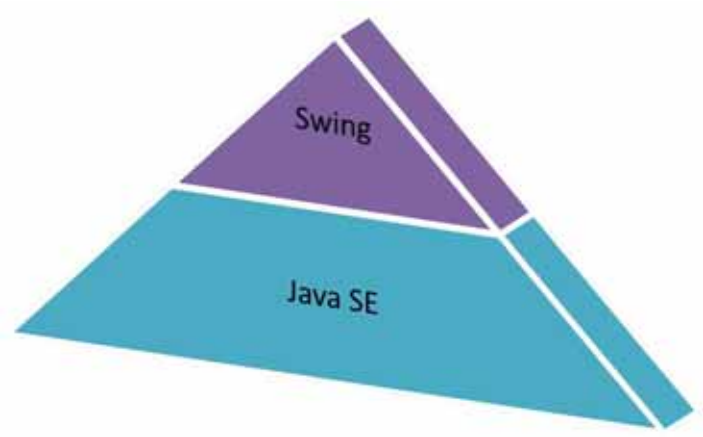

Figura 2. Arquitectura del sistema

2.1.2. Módulos del sistema

El sistema se compone de cinco módulos los cuales se muestran en la Figura 3.

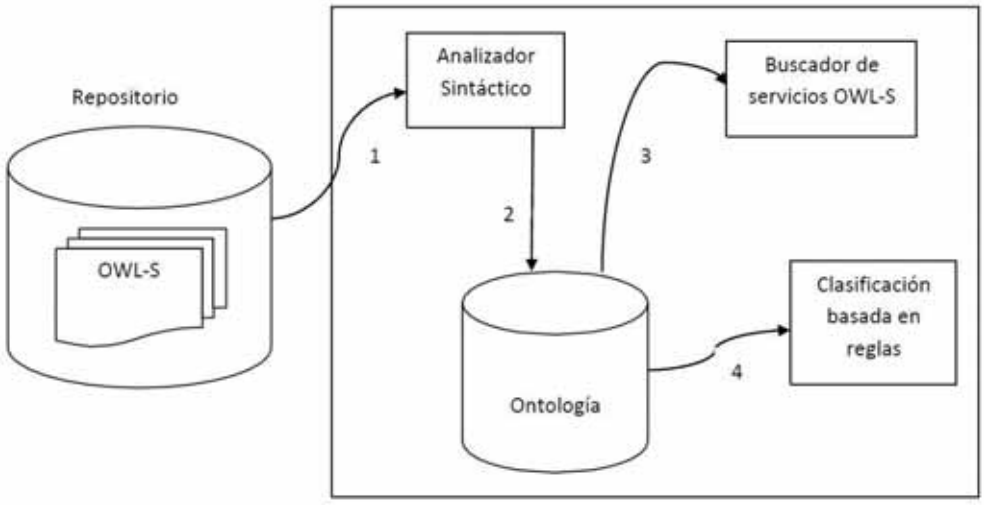

Figura 3. Módulos del sistema

Las funcionalidades de cada módulo son las siguientes:

- **Repositorio de servicios:** Éste módulo almacena todos los servicios web OWL-S para su posterior análisis y clasificación, éste proceso de recolección a almacenamiento de los servicios se contempla que sea manual.
- **Analizador sintáctico de servicios:** Éste módulo se encarga de analizar los servicios OWL-S para obtener los datos necesarios para que sea analizado por la ontología.
- **Ontología de servicios:** Éste módulo se encarga de crear una ontología a partir de los servicios OWL-S que son proporcionados por el analizador sintáctico.
- **Clasificación de servicios:** Éste módulo se encarga de agrupar los servicios que han sido creados en la ontología.
- **Buscador de servicios:** Éste módulo se encarga de realizar consultas de los servicios OWL-S que han sido creados en la ontología.

2.2. Implementación

El proyecto se desarrolló en una pc con las siguientes características:

- Computadora con procesador Intel i7 2600, 8 GB de memoria RAM, disco duro de 1 TB.
- Sistema operativo Windows 7.

2.2.1. Herramientas utilizadas

- **NetBeans**[8]: Es un entorno de desarrollo integrado (IDE), una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. Además es un producto libre y gratuito sin restricciones de uso, también es un IDE de código abierto escrito completamente en Java. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra el *debug* que es una herramienta que sirve para identificar errores durante la ejecución del programa, ya que de otra forma es difícil localizarlos.
- **Protégé 4.1**[9] **:** Es un editor de ontologías gratuito y de código abierto. Ésta plataforma soporta modelado de ontologías mediante un cliente web o un cliente de escritorio. Las ontologías pueden ser desarrolladas en una gran variedad de formatos incluyendo OWL, RDF y XML schema. Este editor está basado en Java y proporciona un entorno *plug and play* que hace que sea una base flexible para la creación rápida de prototipos y desarrollo de aplicaciones.
  Para éste proyecto se utilizó esta herramienta para la creación de la ontología que posteriormente es poblada con los datos que son extraídos por el *parser*. También

fue utilizada para la visualizar gráficamente las instancias que son creadas por el sistema en la ontología.

2.2.2. APIs utilizadas

- **DOM 2.0.5**[10] : El *Document Object Model* (DOM), es API que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido.
  Para este proyecto se utilizó este API para realizar la extracción de los nombres de los servicios OWL-S así como sus nombres y tipos de datos que utiliza cada servicio. Dentro de ésta API, contiene el soporte para *XPath*, que es un lenguaje que permite construir expresiones que recorren y procesan un documento XML, el cual se utilizó para extraer información de nodos dentro de la estructura del documento XML sin tener que recorrer iteradamente todo el documento.
- **Google spelling**[11] **:** Es la API desarrollada por Google, para la revisión ortográfica. En este proyecto fue utilizada para la separación de palabras, dado que se identificó que la gran mayoría de nombre de servicios contenidos en los archivos OWL-S se encontraba sin espacios entre palabras, lo que imposibilitaba realizar el cálculo de la distancia semántica.
- **Gate**[12] **:** Es una API gratuita y de código abierto capaz de solucionar casi cualquier problema de procesamiento de texto, además de tener soporte para la creación y edición de ontologías.
  Para este proyecto se utilizó para el poblado de la ontología que fue creada en Protegé a partir de los datos que fueron extraídos por medio del *parser.*
- **RITa.WordNet**[13] **:** Es una API que proporciona un fácil acceso a la otología de WordNet. WordNet es una gran base de datos léxica de inglés. Esta base de datos proveé Sustantivos, verbos, adjetivos y adverbios que se agrupan en conjuntos de sinónimos cognitivos (*synsets*), cada una expresando un concepto distinto. Los *synsets* están vinculados entre sí mediante relaciones conceptuales, semánticas y léxicos. La estructura de WordNet es una herramienta útil para la lingüística computacional y procesamiento del lenguaje natural.
  Para éste proyecto se utilizó ésta API para obtener las distancias semánticas que guardan entre dos palabras, específicamente, entre los nombres de servicios y de parámetros de entrada/salida para así poder establecer una relación de agrupamiento entre servicios para así realizar la clasificación de éstos.

2.2.3. Módulos del sistema

- **Repositorio de servicios:** Para el desarrollo del sistema como para la fase de pruebas, se utilizaron una colección de servicios que fueron descargados de la página *SemWebCentral*[1] .
  Al analizar los tipos de servicios se encontró que en dicha colección existen dos tipos de versiones de servicios OWL-S, la versión 1.0, y la 1.1. La principal diferencia entre los dos tipos de versiones consiste en que en la versión 1.1 los parámetros de entrada/salida están descritos mediante el atributo *rdf:daraType*, mientras que los de la versión 1.0 está especificada por el atributo *rdf:resource.*
  Para la mayoría de los servicios contenidos se encontraban en ambas versiones con lo cual fue posible probar el sistema bajo éstas versiones.
  También se identificó que la mayoría de los nombres de servicios, como nombres de parámetros entrada/salida se encontraban no se encontraban separados por espacios entre palabras lo cual sólo era legible para un humano pero impedía el procesamiento automatizado.
- **Analizador sintáctico de servicios:** El analizador sintáctico o *parser,* fue desarrollado utilizando el API DOM y el lenguaje XPath con lo cual se optimizó las búsquedas de los datos que se quieren extraer sin tener que realizar una búsqueda exhaustiva a través de toda la estructura que se está analizando.
  También se implementaron hilos en el proceso del análisis sintáctico, la cual mediante una expresión XPath se obtiene el nodo en el cual está la información a extraer, el procesamiento multihilos se encarga de extraer la información que está contenida en nodos hijos o en atributos dentro del nodo. También se desarrolló un método el cual hace la diferenciación entre versiones de los servicios OWL-S para así poder realizar el procesamiento.
- **Ontología de servicios:** En ésta fase se creó una ontología la cual contendría todos los servicios que fueron procesados por el analizador sintáctico. Ésta ontología fue creada con el editor Protegé, ya que de manera gráfica se pueden elegir las clases como relaciones que son necesarias para cada ontología a crear. La estructura que tiene ésta ontología se muestra en la Figura 4.
  Una vez creada la ontología se guarda ésta en el formato OWL, y el sistema lee el archivo para crear instancias de las clases que están contenidas en la ontología y así poder realizar el poblado de datos. En esta fase se desarrolló una clase Java la cual mediante el API de Google Spelling se pudieron separar los nombres de servicios y nombres de parámetros de entrada/salida para poder así guardarlos en la ontología y posteriormente su procesamiento semántico para la clasificación.
  Para el poblado de la ontología de utilizó el API Gate, con el cual se crearon las instancias y las relaciones entre clases con los datos que fueron extraídos de los servicios. También se identificó que muchos servicios tenían nombres de parámetros de entrada/salida idénticos, con lo cual no se puede crear dos instancias con el mismo nombre, lo cual se solucionó añadiendo una cadena predefinida las veces que fuera necesario al encontrar el nombre de un parámetro idéntico a uno ya procesado.
- **Clasificación de servicios:** En esta fase se desarrolló una clase Java para el cálculo de la distancia semántica entre los nombres de los servicios y cada uno de los tipos

de clasificación. Para realizar éste proceso se eligieron los nombres de los nombres de las clasificaciones teniendo en cuenta previamente la temática de los servicios a procesar; se pueden elegir el número de categorías que se quiera, sin embargo se debe de tener en cuenta que el tiempo de tiempo de clasificación va a ser proporcional al número de categorías que se hallan elegido. Para el nombrado de las categorías se seleccionó sólo una palabra que fuera distintiva de la temática que se quería abarcar dentro de la clasificación, sin embargo, también es posible nombrar una clasificación con más de una palabra, pero así como el número de clasificaciones afecta al tiempo de clasificación, el número de palabras que formen el nombre de cada una de las clasificaciones va ser proporcional al tiempo en que se tardará dicho proceso.
Una vez que se ha concluido el proceso de clasificación se escriben las los datos de todos los servicios ya clasificados en su categoría en un archivo de texto plano para que pueda ser leído por el buscador de servicios y no se tenga que hacer el procesamiento de los servicios cada vez que se realice una búsqueda. También se crean carpetas con el nombre de cada categoría y se organizan físicamente los archivos que contienen los servicios en su respectiva carpeta.

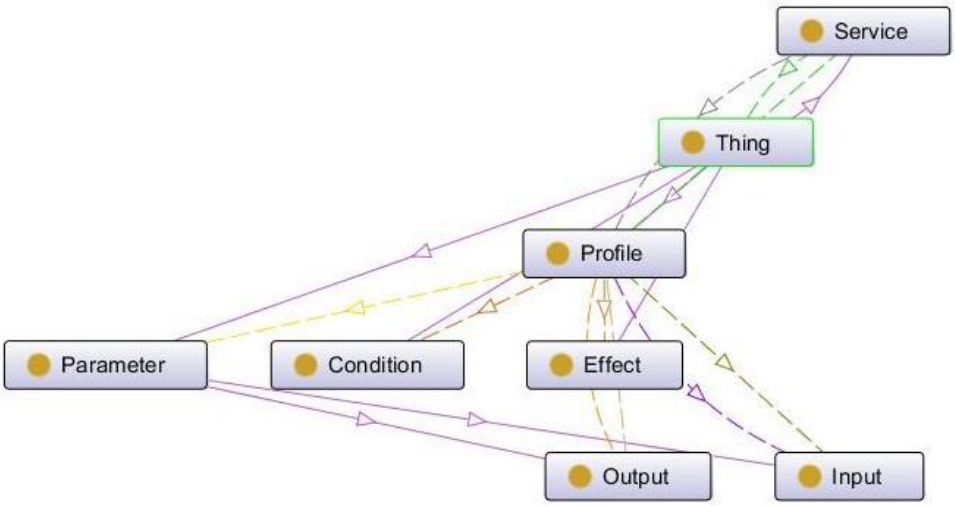

Figura 4. Estructura ontológica del sistema

2.2.4. Algoritmo de clasificación

El algoritmo utilizado para la clasificación de los servicios OWL consiste en que teniendo un conjunto de nombres de servicios calcular la distancia semántica entre cada uno de éstos nombres con un conjunto de nombres que corresponden a la descripción general de cada clasificación y agruparlos de acuerdo a un rango de similitud previamente establecido.

2.2.4.1. Algoritmo

Teniendo un vector de nombres de clasificaciones A y un arreglo bidimensional B el cual contiene un conjunto N de nombres de servicios, el peso más bajo obtenido P y la clasificación C a la cual pertenece el servicio. Sea D el vector de distancias y x el primer elemento del arreglo bidimensional.

1. Se toma el nombre del servicio, B(x,0), y se agregan espacios entre palabras en caso de que el nombre del servicio no cuente con éstos.
2. Se divide B(x,0) en palabras y se almacenan en un vector S.
3. Para cada elemento de S se calcula la distancia semántica contra cada elemento de A, se realiza la suma de todos las distancias del vector S contra cada elemento de A entre el tamaño del vector A y se almacena dicho cálculo en el vector de distancias D.
4. Se toma el índice I correspondiente de la distancia más cercana a 0 del vector D, lo cual indica que el nombre de dicho servicio es lo más parecido a la clasificación almacenada en el índice I del vector A.
5. Se almacena en B(x,1) el valor de la mejor distancia obtenida.
6. Se almacena en B(x,2) el índice I correspondiente a la mejor distancia obtenida.
7. Se incrementa en uno x y se repite el proceso desde 1 hasta el último elemento de B.

2.2.4.2. Pseudocódigo

Sea B un arreglo bidimensional B el cual contiene un conjunto N de nombres de servicios y un vector de nombres de clasificaciones A.

```
algoritmo (B, A)
    para i = 0 hasta N hacer
        B(i,0) = agregar_espacios(B(i,0))
        S = dividir_en_palabras(B(i,0))
        para j = 0 hasta tamaño de A hacer
            para k = 0 hasta tamaño de S hacer
                total = 0
                total = total + distancia_semantica(S(k), A(j))
                D(j) = total / tamaño de A
        x = INFINITO
        para j = 0 hasta tamaño de D hacer
            si x < D(j) entonces
                x = D(j)
                I = j
        B(i,1) = x
        B(i,2) = I
```

3. Pruebas del sistema

Las pruebas al sistema se realizaron en una computadora con un procesador AMD Phenom II B77 a 3.2 GHz y 4 GB de memoria RAM. Para la extracción de servicios OWL-S se utilizó la colección que fue descargada de la página *SemWebCentral.* El número de servicios OWL-S de la versión 1.0 con la que fue probado el sistema fue de 556, mientras que para la versión 1.1 fue de 1, 083 servicios.

- **Pantalla inicial.** La pantalla inicial consiste en un componente en el cual se proporciona el directorio en donde se alojan los servicios OWL-S a procesar, un buscador de servicios, el cual está disponible una vez que se haya hecho el procesamiento de los servicios o bien, servicios que hayan sido procesados anteriormente y una opción para descargar la ontología resultante del procesamiento.

Figura 5. Pantalla inicial del sistema

Las pruebas que se ejecutaron para ésta pantalla fueron las siguientes:

- Se proporcionó un directorio que no contenía ningún archivo OWL-S, y el resultado fue que se desplegó un mensaje de error indicando que se debe de proporcionar un directorio en donde exista al menos un servicio OWL-S (*véase* Figura 6).

Figura 6. Carga de directorio inválido

- Se proporcionó un directorio que contenía archivos con extensión .owls y además otros archivos con otras extensiones distintas a la anterior, y el resultado fue que el sistema solamente cargo los archivos con extensión .owls e ignoró los demás archivos.
- Se proporcionó un directorio que contenía solo archivos con extensión .owls, y el resultado fue que el sistema cargó todos los archivos
- Se proporcionó un directorio vacío, y el resultado fue que el sistema mostró un mensaje error indicando que se debe de proporcionar un directorio que contenga servicios OWL-S (*véase* Figura 1)
- Se eligió la opción *Cargar* sin haber proporcionado ningún directorio, y el resultado fue que el sistema mostró un mensaje de error indicando que se debe proporcionar un directorio que contenga servicios OWL-S (*véase* Figura 1)

- **Procesamiento de servicios.** El procesamiento de servicios consiste en la extracción de los datos relevantes de loa archivos OWL-S, clasificación y organización de los archivos en directorios físicamente.

  Las pruebas que se ejecutaron es ésta fase fueron las siguientes:

  - Se proporcionó un directorio con 556 archivos con extensión .owls correspondientes a la versión 1.0, el resultado fue que la pantalla mostró un indicador de proceso (*véase* Figura 7) terminando exitosamente y creando directorios físicamente con las categorías definidas en el sistema (véase. Figura 8). El tiempo que tardó el sistema en realizar el procesamiento fue de 3.7 minutos.

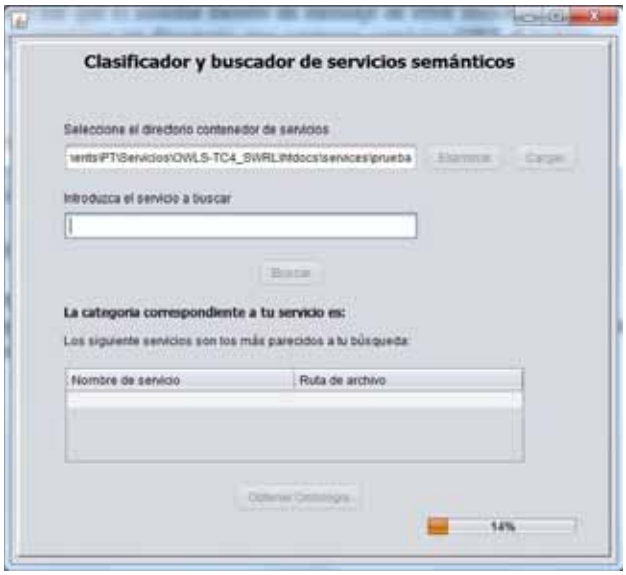

Figura 7. Procesamiento de servicios

- Se proporcionó un directorio con 1,083 archivos con extensión .owls correspondientes a la versión 1.1, el resultado fue que la pantalla mostró un indicador de proceso (*véase* Figura 7) terminando exitosamente y creando directorios físicamente con las categorías definidas en el sistema (véase. Figura 8). El tiempo que tardó el sistema en realizar el procesamiento fue de 8.5 minutos

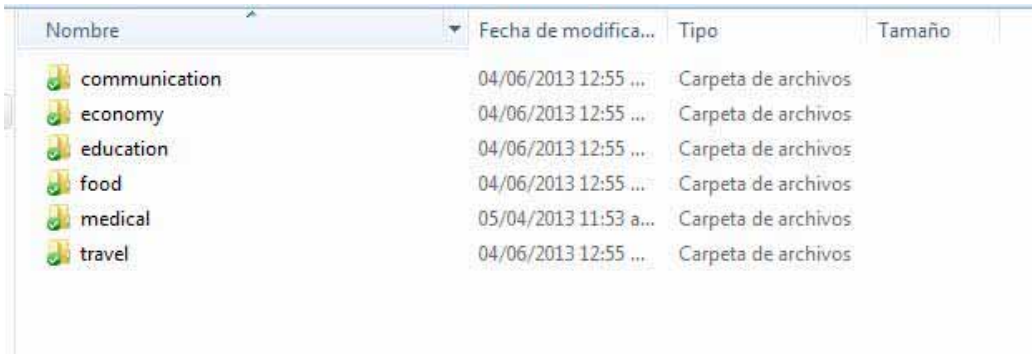

Figura 8. Directorios clasificados

- **Poblado de la ontología.** Consiste en crear instancias de las clases de la ontología a partir de los datos extraídos de los servicios OWL-S y agregarlos a la ontología inicial y exportar el archivo.

  Las pruebas que se ejecutaron en ésta fase fueron las siguientes:

  - Se eligió la opción *Obtener ontología* procesado servicios OWL-S anteriormente, y el resultado fue que el sistema generó un archivo de salida llamado *Salida.owl* (*véase*. Figura 9) con ninguna instancia creada.

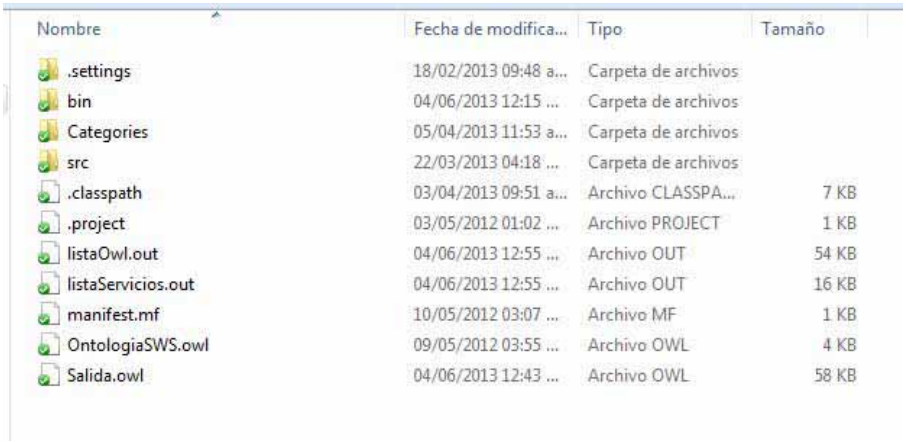

Figura 9. Ontología resultante

- Se eligió la opción *Obtener ontología* procesado servicios OWL-S anteriormente, y el resultado fue que el sistema generó un archivo de salida llamado *Salida.owl* (*véase*. Figura 9) con las instancias creadas a partir de los datos extraídos de los servicios OWL-S

- **Buscador de servicios:** Este buscador proporciona una lista de nombres servicios y ruta física en donde se encuentran, además de indicar a qué categoría es la más probable que pertenezca a partir de la búsqueda ingresada.

  Las pruebas que se ejecutaron en esta fase fueron las siguientes:

  - Se seleccionó la opción *Buscar* sin haber ingresado ningún dato en el campo de búsqueda, y el resultado fue que el sistema mostró un mensaje de error indicando que debe de proporcionar una dato para realizar la búsqueda (véase. Figura 10).

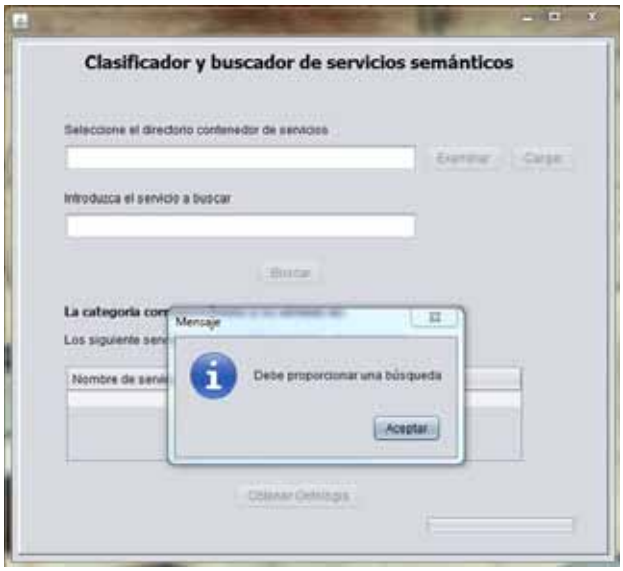

Figura 10. Mensaje de error en búsqueda

- Se proporcionó un búsqueda sin haber realizado un procesamiento de servicios OWL-S anteriormente, y el resultado fue que el sistema mostró un mensaje de error indicando que sebe de proporcionar un directorio de servicios antes de realizar una búsqueda (véase. Figura 11).

Figura 11. Búsqueda sin servicios existentes

- Se proporcionó una búsqueda habiendo realizado un procesamiento de servicios OWL-S anteriormente, y el resultado fue que el sistema mostró el nombre de la categoría al cual se asemeja dicha búsqueda y una lista de los servicios contenidos en dicha categoría.

Figura 12. Búsqueda exitosa

- **Vista de la ontología en Protegé:** Mediante el uso de éste programa se puede visualizar las instancias de las clases que se generaron a través del sistema, así como el esquema gráfico de la ontología.

    - **Vista de entidades.**

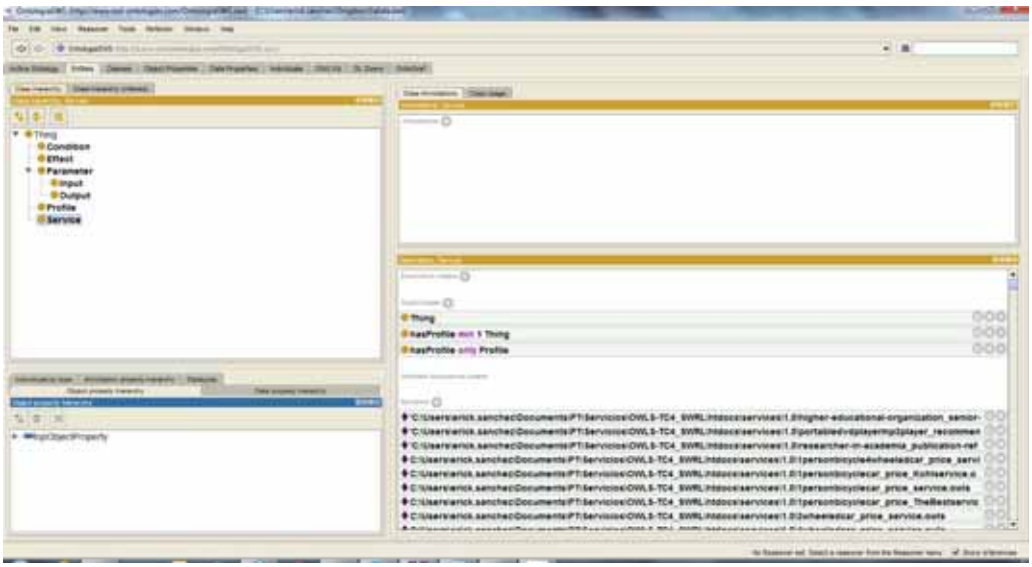

Figura 13. Vista de entidades

    - **Vista de individuos.**

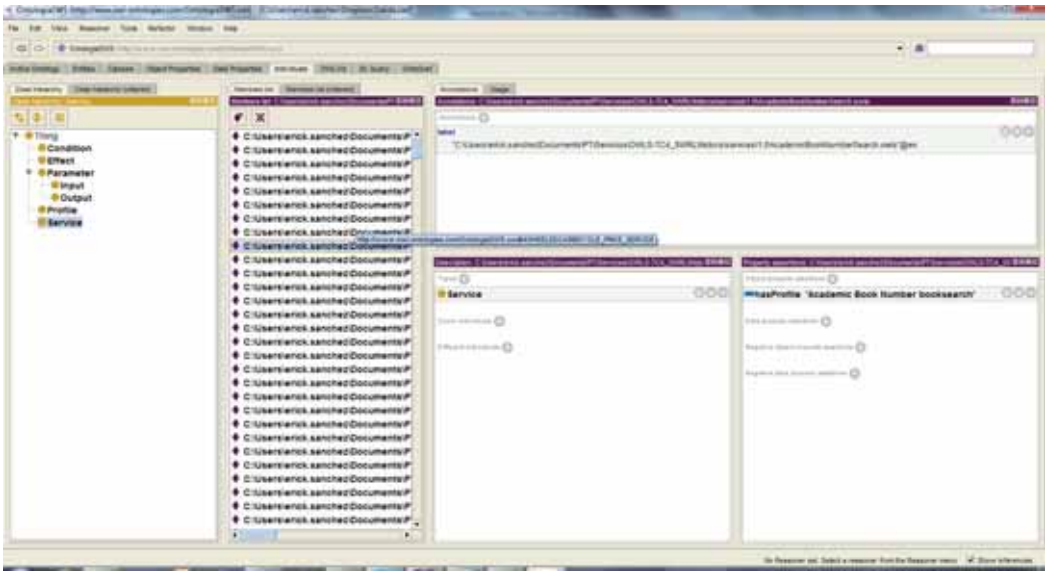

Figura 14. Vista de individuos

- **Vista gráfica de la ontología.**

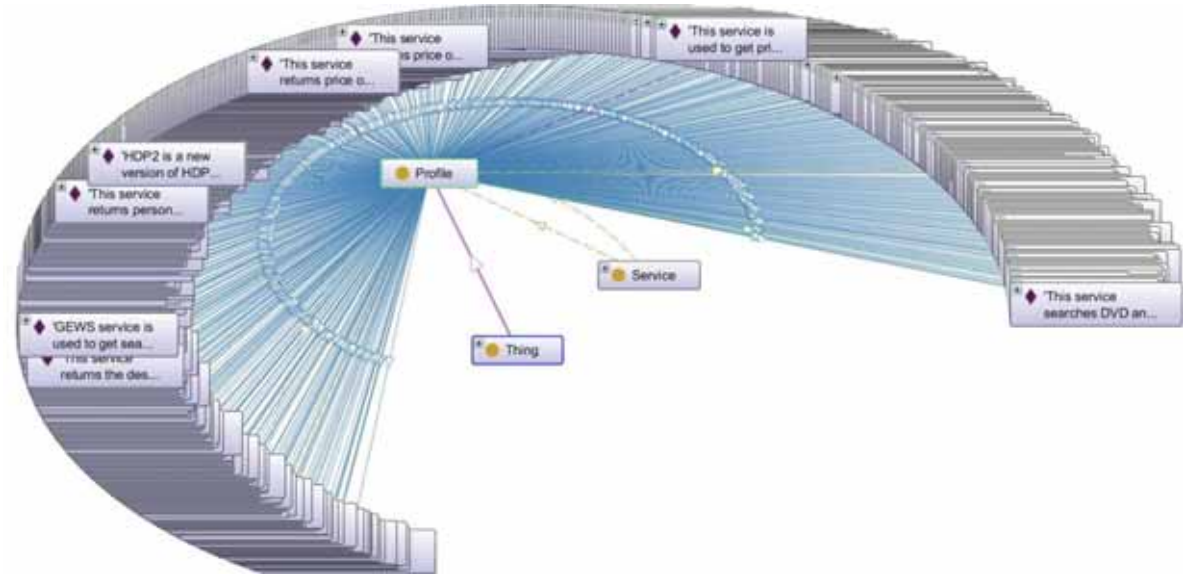

Figura 15. Vista gráfica de la ontología

4. Conclusiones

Al finalizar éste proyecto terminal he tenido un panorama más amplio sobre los servicios OWL-S así como el trabajo de las ontologías en cuanto al diseño e implementación.

Con el estudio y desarrollo de éste sistema pude ver la importancia de las ontologías en la web semántica, ya que es sabido que las ontologías es el fundamento y base para el funcionamiento de ésta, me fue necesario poderme haber adentrado a conocer un poco cómo funcionan las ontologías y la importancia que tienen en ésta web semántica.

En cuanto a la creación de una ontología, creo y por experiencia, que la fase de análisis de diseño de ésta es determinante para que un sistema tenga una base de conocimiento que nos sea funcional, de lo contrario los resultados que obtendremos no serán los esperados, es por eso que a mi parecer la parte más compleja al desarrollar un sistema basado en conocimiento es analizar ampliamente la funcionalidad deseada para poder diseñar una ontología de acuerdo a éstas necesidades.

Por otra parte, el estudio de los servicios OWL-S, me permitió conocer y trabajar con otro tipo de servicios además de los más conocidos que son los servicios web, aunque al igual que los antes mencionados, me encontré que existen versiones de éstos y que al extraer la información de sus descriptores XML no se podía dar el mismo tratamiento a éstos.

La mayor dificultad durante el desarrollo del proyecto, fue que en la mayoría de los nombres de los servicios se encontraban nombrados sin espacios lo que resultó, en primera instancia, imposible de realizar su cálculo semántico, ésta situación aparentemente sólo podía ser reconocida por un humano, que en éste caso tuviera un conocimiento de palabras en inglés ya que éstos nombres de servicios están en éste idioma. En un principio, pensé en solucionar éste problema procesando la concatenación secuencial de letras, es decir, tomar la primera las primeras dos letras y compararla contra un diccionario de palabras en inglés y mediante una comparación sintáctica poder saber si se trataba de una palabra existente o no, en caso que no existiera concatenar la siguiente letra y realizar el mismo proceso hasta que hubiera una ocurrencia de una palabra en inglés; sin embargo es bastante obvio que el tiempo de procesamiento sería inmenso y los más seguro, no sería funcional, no obstante ésta idea nada eficiente me llevó a pensar en usar un analizador ortográfico (spell checker), lo que me llevó a recordar la funcionalidad que la mayoría de nosotros nos ha pasado en más de una ocasión que al usar el navegador de Google y al escribir sintácticamente incorrecta una palabra o conjunto de palabras, éste nos sugiere un una opción a lo que probablemente se quiso escribir, y al realizar la prueba con un conjunto de palabras que no se encontraban separadas con espacios, encontré que también sugería una opción a lo que se había ingresado, siendo el resultado la frase ya separada por espacios, además si se encontraba algún error ortográfico en alguna palabra también es corregida, por lo que el utilizar el API de google spell checker fue una solución exacta, además de que ésta API es de código libre y gratuita, sin embargo, el único inconveniente, si así lo podemos llamar, es la dependencia de una conexión a internet y requerir que el servicio de éste proveedor esté activo, además de que el tiempo de procesamiento de los servicios OWL-S van a estar sujetos al tiempo de petición al servidor de éste corrector ortográfico, pero por mi parte creo que es una solución suficientemente buena y funcional.

Alguna de las mejores ayudas, en cuanto a API's, fue la de Gate, ésta me resultó de gran utilidad, ya que para el tratamiento de ontologías resulta sumamente fácil y talmente funcional el trabajar con esta API, además que esta API contiene un gran soporte para el procesamiento del lenguaje natural, aunque en éste proyecto no fue utilizado.

Otra API que fue de mucha utilidad fue la de RITa.WordNet, que es una implementación de la API de WordNet, lo cual me ayudo en gran manera a realizar el cálculo semántico de las palabras para así determinar grupos o clasificaciones de los servicios que son procesados.

En cuanto a los resultados esperados contra los resultados obtenidos encontré que para poder obtener un mejor resultado de clasificación es de suma importancia conocer con precisión la temática de los servicios que se deseen agrupar, y una vez conociendo esto es importante hacer un análisis semántico detallado para escoger los nombres de las clasificaciones ya que de esto depende si los nombres de los servicios serán bien clasificados o no.

Para éste proyecto, los resultados obtenidos fueron aceptables tomando en cuenta que el cálculo de la distancia semántica fue obtenido por medio del API de WordNet y que no se realizó un análisis exhaustivo para la elección de los nombres de las clasificaciones. EN específico pude notar que la mejor categoría fue la de *Food* (Comida), ésta categoría resultó ser muy descriptiva y abarcar la mayor cantidad de servicios que en realidad pertenecen a la temática de servicios de comida, mientras que la peor categoría resultó ser la de *education* (educación) que abarcó muchos servicios que no pertenecían a ésta temática, lo que explica que la elección de correctos nombres de clasificación cambia considerablemente los resultados.

Anexo

- **Diagrama de clases**

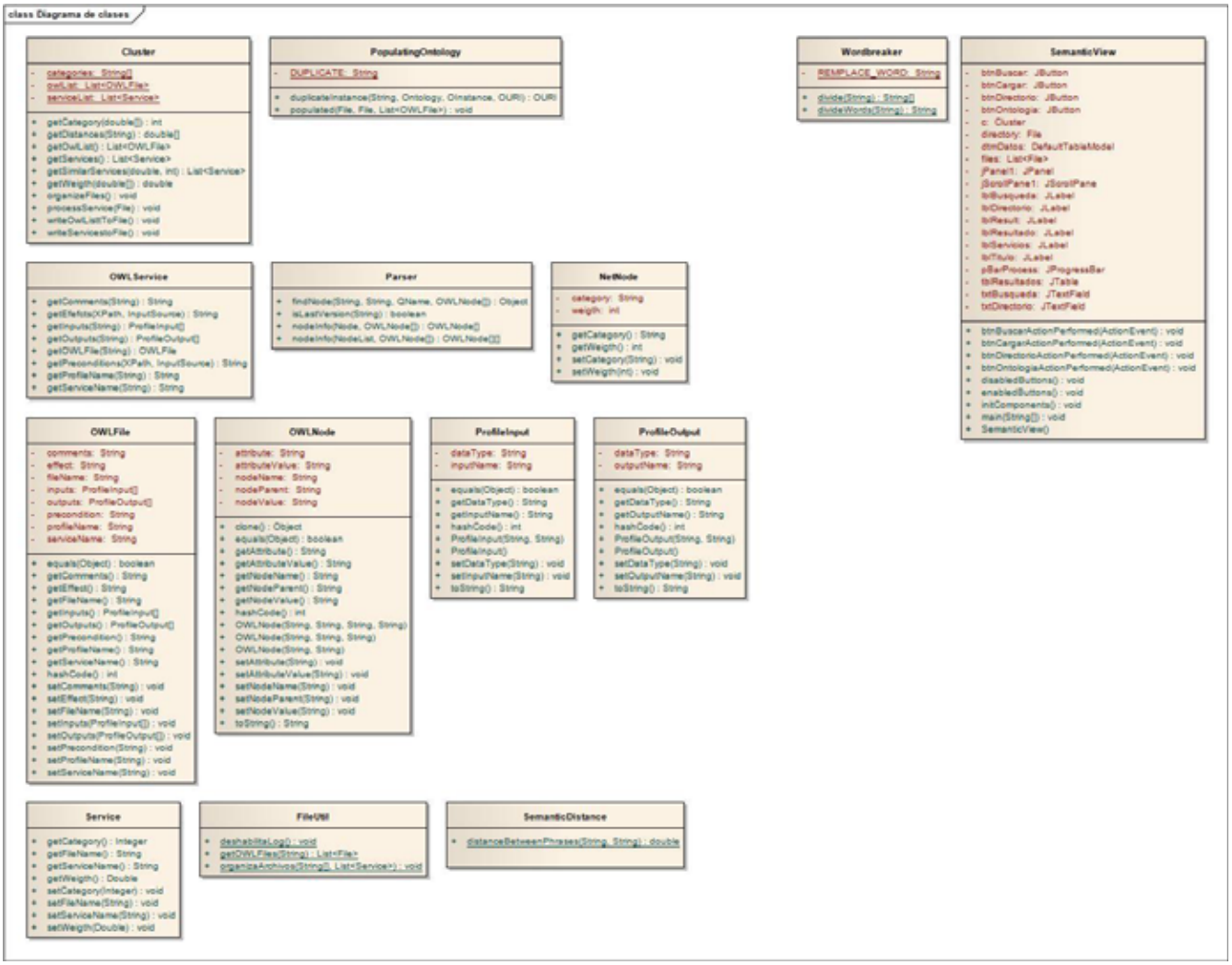

Figura 16. Clases del sistema

Referencias

[1] *SemWebCentral* (2005, Abril) [En línea]. Disponible: http://projects.semwebcentral.org/projects/owls-tc/

[2] J. P. Martínez. "*Extracción automatizada y representación de servicios Web mediante ontologías*", propuesta de proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2011.

[3] D. M. Mercado. "*Solución de problemas de mediana dificultad utilizando composición de servicios web*", propuesta de proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F. México, 2011.

[4] L. A. Toral. "*Modelo Multidimensional para la representación de Perfiles de Aprendizaje y Estilos de Pensamiento mediante Ontologías y Reglas de Inferencia*", propuesta de proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2011;

[5] *OWL-S: Semantic Markup for Web Services* (2004, Noviembre 22) [En línea]. Disponible: http://www.w3.org/Submission/OWL-S/

[6] *OWL-S 1.0 Release* (2012, Marzo) [En línea]. Disponible: http://www.daml.org/services/owl-s/1.0/

[7] *Java SE* (2013, Junio) [En línea]. Disponible: http://www.oracle.com/technetwork/es/java/javase/overview/index.html

[8] *Netbeans* (2013, Marzo) [En línea]. Disponible: https://netbeans.org/

[9] *Protégé* (2013, Abril) [En línea]. Disponible: http://protege.stanford.edu/

[10] *Document Object Model (DOM)* (2005, Enero) [En línea]. Disponible: http://www.w3.org/DOM/

[11] *Google Spelling* (2010, Abril) [En línea]. Disponible: http://code.google.com/p/google-api-spelling-java/

[12] *Gate* (2012, Noviembre) [En línea]. Disponible: http://gate.ac.uk/download/

[13] *RiTa.WordNet* (2012, Septiembre) [En línea]. Disponible: http://www.rednoise.org/rita/wordnet/documentation/

UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD AZCAPOTZALCO

DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

Clasificación de servicios web semánticos mediante ontologías

MANUAL DE USUARIO

Alumno
Erick Sánchez Estrada

Asesora
Dra. Maricela Claudia Bravo Contreras
Departamento de Sistemas

Abril 2013

Tabla de contenido

1. Interfaz del sistema

Para acceder a la aplicación debe abrir, desde cualquier sistema operativo, una terminal o símbolo del sistema e ingresar el siguiente comando *java –jar Clasificador.jar* como se muestra en la Figura 1.

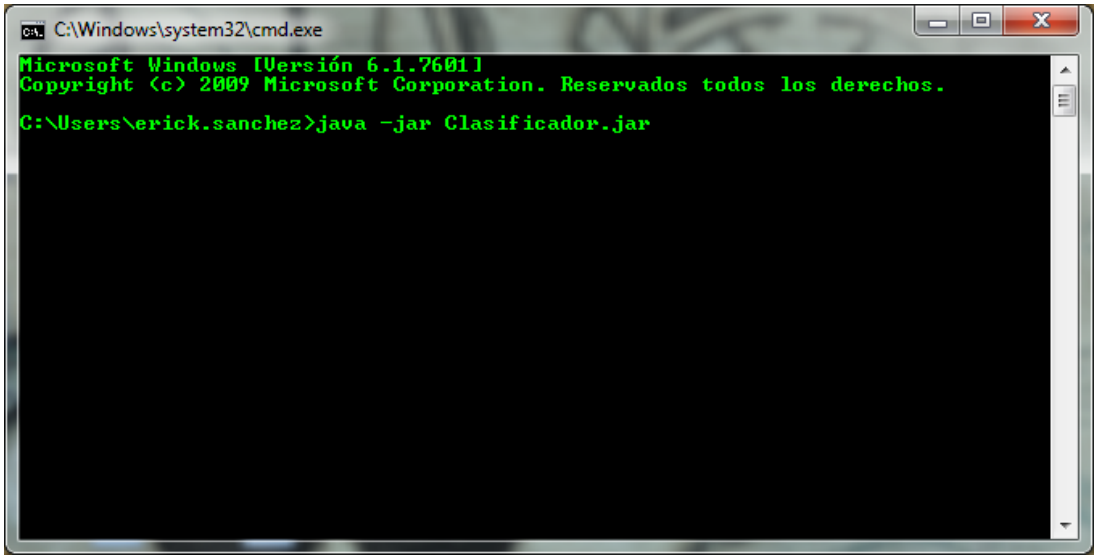

Figura 1. Terminal

Luego se mostrará la pantalla inicial del sistema como se muestra en la Figura 2

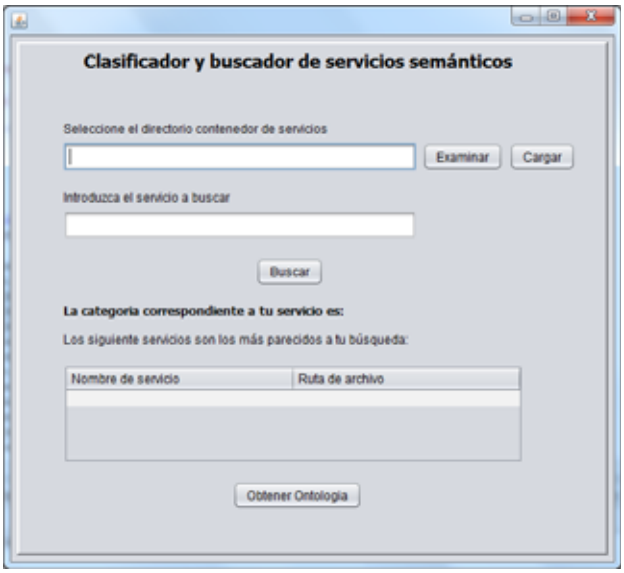

Figura 2. Pantalla principal

2. Seleccionar directorio de servicios OWL-S

Para realizar la carga de los servicios OWL-S, se selecciona el botón *Examinar* y se mostrará un selector de archivos en el cual se escogerá el directorio contenedor de dichos servicios como se muestra en la Figura 3.

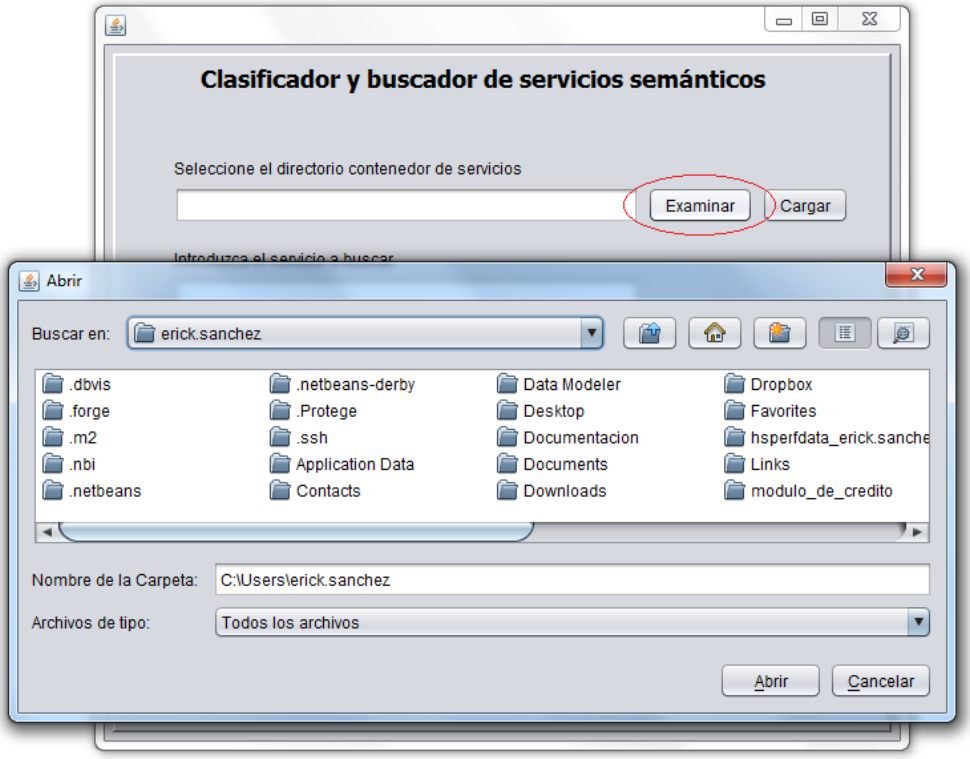

Figura 3. Selector de directorio

3. Carga y procesamiento de los servicios

Una vez seleccionado el directorio contenedor de los servicios OWL-S se selecciona el botón *Cargar* y en la pantalla aparecerá un indicador de progreso como se muestra en la Figura 4.

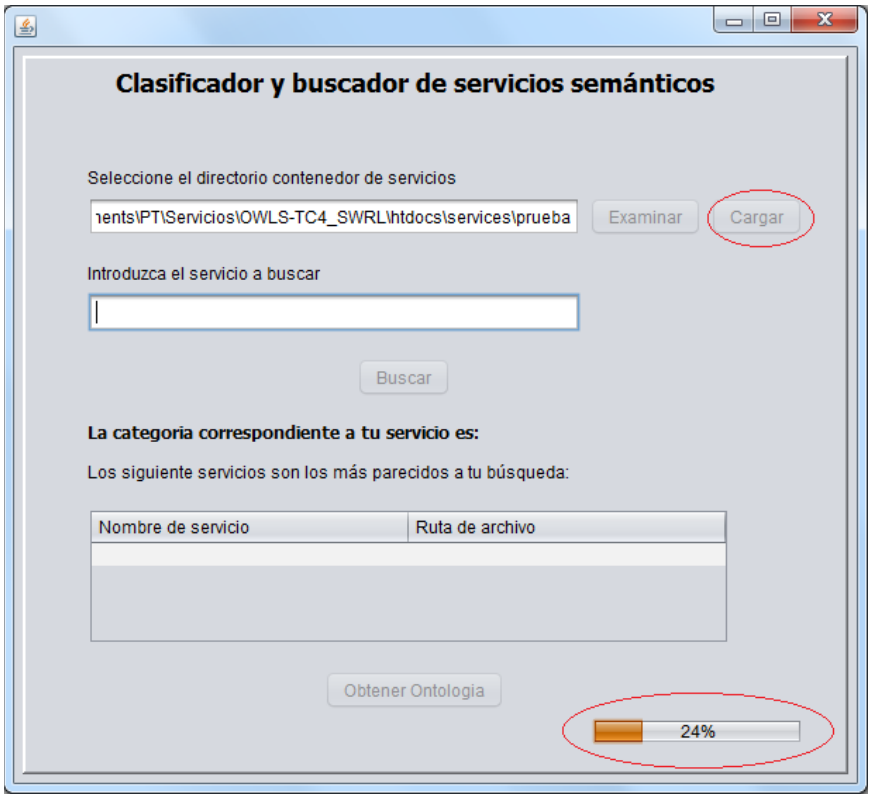

Figura 4. Carga y procesamiento de servicios

4. Generación de ontología resultante

Una vez terminado la carga y procesamiento de los servicios OWL-S, se puede elegir obtener la ontología resultante de la clasificación seleccionando el botón *Obtener Ontología* como se muestra en la Figura 5, lo cual generará el archivo *Salida.owl* en el directorio en donde se ejecutó el sistema.

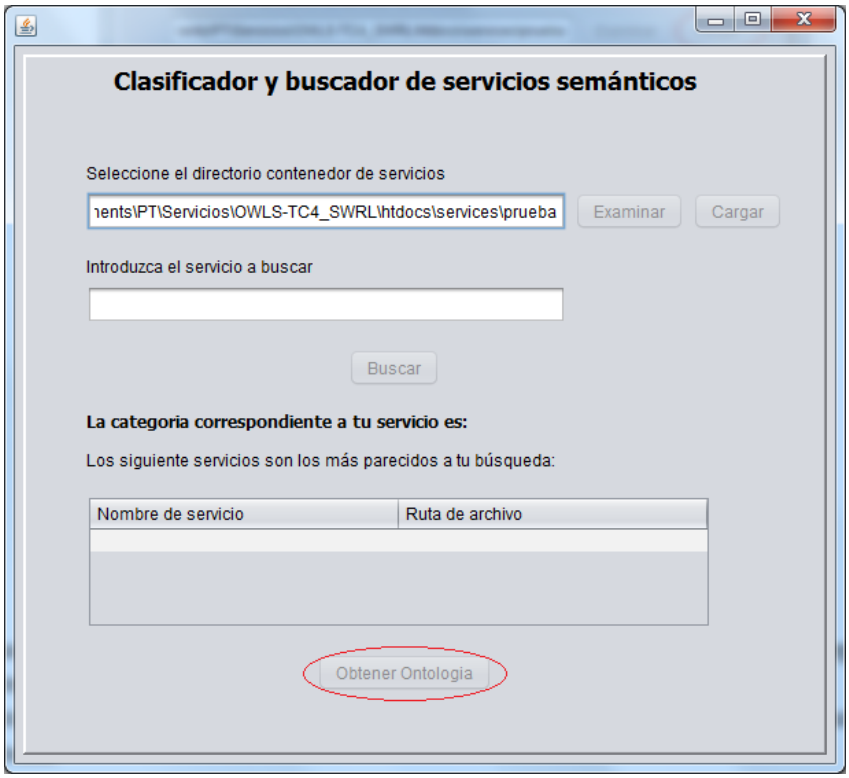

Figura 5. Obtener ontología

Una vez generado la ontología se puede visualizar los individuos (*véase* Figura 6) contenidos en ésta, y también se puede visualizar gráficamente la ontología (véase Figura 7).

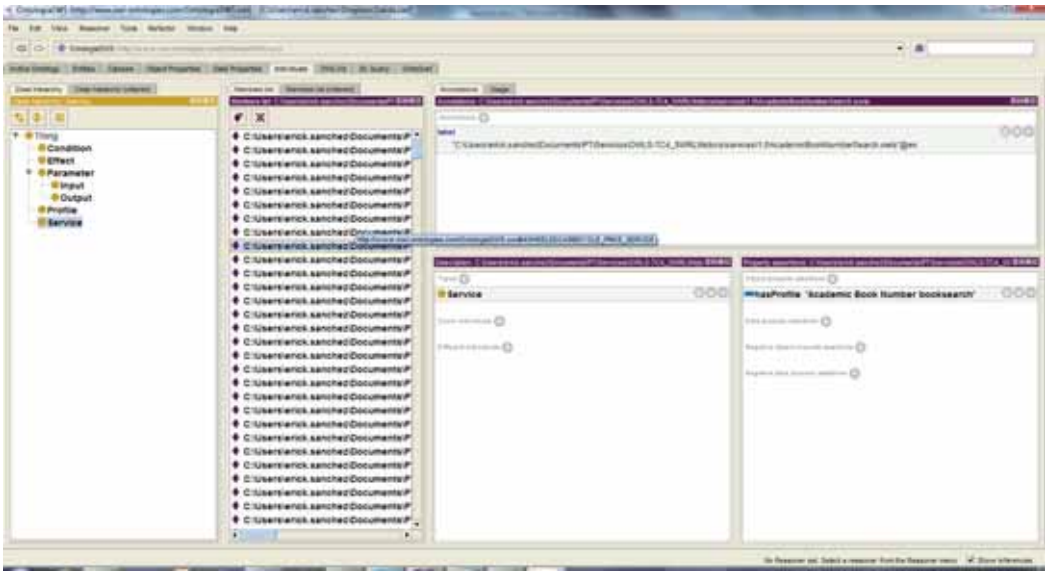

Figura 6. Vista de individuos

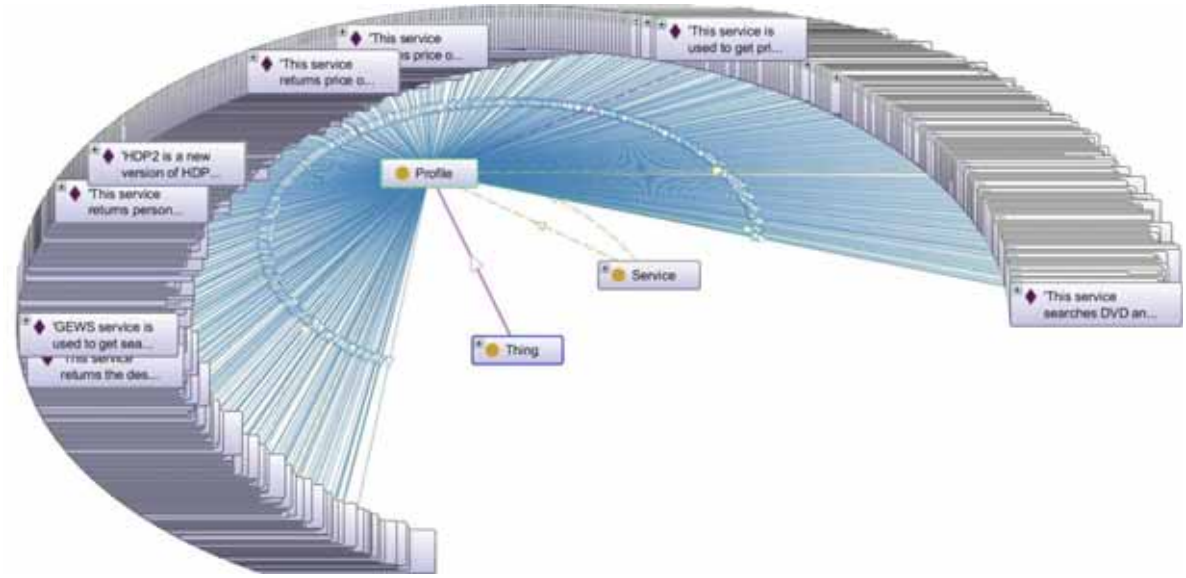

Figura 7. Vista gráfica de la ontología

5. Búsqueda de servicios

Para realizar una búsqueda, se ingresa una palabra o un conjunto de palabras que hagan referencia a la temática de una de las clasificaciones y se selecciona el botón Buscar y el resultado será que el sistema muestra el nombre de la clasificación a la cual hace más referencia dicha búsqueda y una lista de los nombres y ruta física de los servicios que están contenidos en dicha clasificación como lo muestra en la Figura 8.

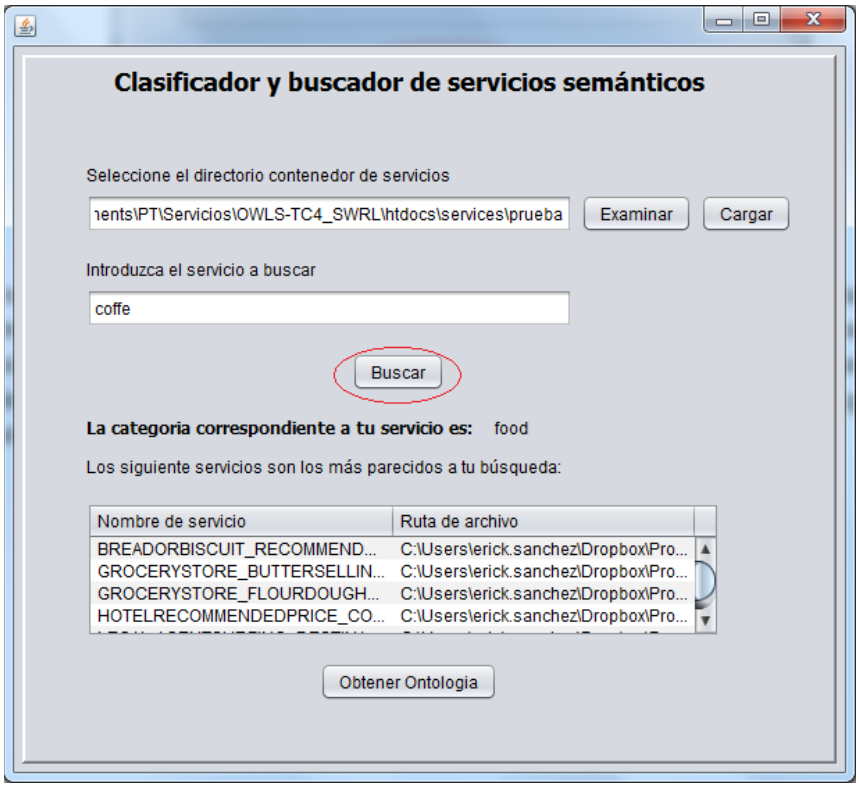

Figura 8. Búsqueda