Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Asistente para la evaluación de la eficiencia lógica y coherencia terminológica de ontologías públicas

Pedro Antonio Silva Sánchez
208205347

Trimestre 2014 Primavera

Julio de 2014

Asesor:

Dra. Maricela Claudia Bravo Contreras
Profesor Asociado
Departamento de Sistemas

Yo, Maricela Claudia Bravo Contreras, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación de la Biblioteca Digital, así como el Repositorio Institucional de UAM Azcapotzalco.

Firma

Yo, Pedro Antonio Silva Sanchez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como el Repositorio Institucional de UAM Azcapotzalco.

Firma

Índice de Contenido

1. Resumen.

En la historia de la computación ha existido una cantidad finita de información que al paso de la historia se ha ido modificando a tal grado de optimizarla y sacar el máximo provecho de dicha información, tal es así como la historia de la creación de la tierra que al paso del tiempo se ha ido modificando la distintas teorías puesto que las investigaciones obligan a investigar más sobre las posibles causas.

Tal caso es así que en la web existen diferentes tipos de herramientas que nos ayudan a recabar una infinita cantidad de información que con la optimización de programas nos facilitan el manejo de estas inmensas bases de datos, tal caso son los servicios Web que nacen a raíz de estas grandes bases de datos, los cuales se denominan ontologías.

La realización de este proyecto terminal permite la obtención de una secuencia de propiedades de cada una de las ontologías cargadas en un formato OWL únicamente se basa en este formato, puesto que se mide la coherencia y la terminología de estas mismas, mediante un orden cronológico respecto a la cantidad del tamaño de dicho archivo, si bien esto facilita al usuario solo trabajar con ontologías coherentes.

2. Introducción

En este proyecto terminal se involucran los conceptos de eficiencia lógica y coherencia terminológica para la evaluación de un conjunto de ontologías públicas.

Una ontología se define como una especificación formal de una conceptualización compartida [1]. Es decir un marco común o una estructura conceptual sistematizada y de consenso no solo para almacenarla, sino también para buscarla y recuperarla.

La eficiencia lógica de una ontología se puede calcular midiendo el tiempo que tarda en responder a una consulta lógica sobre los conceptos y las relaciones que se encuentran almacenados en dicha ontología. El diseño de una ontología tiene un efecto directo sobre la eficiencia lógica de la ontología, una ontología mal diseñada presenta problemas de eficiencia lógica. De tal forma que la eficiencia lógica es un indicador importante que permite evaluar la calidad del diseño interno de la ontología. Para evaluar la eficiencia lógica de las ontologías se ejecutarán un conjunto definido de consultas a las diferentes ontologías y se obtendrá el número de iteraciones que lleva en responder a las consultas, de tal forma que se pueda observar las diferencias concretas.

La coherencia terminológica permite revisar si la ontología no presenta contradicciones axiomáticas en las definiciones y relaciones.

Concretamente para este proyecto terminal se propone diseñar e implementar una herramienta que realice una evaluación de la eficiencia lógica y la coherencia terminológica de ontologías públicas.

3. Antecedentes

3.1.1 Trabajos relacionados.

3.1.2 Proyectos terminales.

1. - *Sistema clasificador de ontologías mediante métodos de máquinas de soporte vectorial* [2]. Este proyecto trabaja con ontologías para hacer la clasificación de los buscadores en la web, pero no calcula el tiempo en la evolución lógica que tarda en consultar una ontología.

2. - *Sistema de Recuperación de Información de Textos de Investigación de la Web* [3]. En este proyecto realiza una búsqueda con palabras claves para buscar y recuperar los documentos relacionados, pero en caso de encontrar una ambigüedad no realiza la coherencia terminológica y no calcula la eficiencia lógica.

3. - *Sistema semántico para la representación de contextos utilizados en aplicaciones de computo obicuo* [4]. En este proyecto realiza una evaluación en las ontologías de ciertas características pero no se muestran las ambigüedades del resultado de la coherencia terminológica.

3.2 Artículos

4. – *Mapeo de Ontologías Orientado a la Eficiencia* [4]. En este artículo realiza una comparación en términos de eficiencia y de calidad de los resultados obtenidos en las ontologías, pero no la coherencia terminológica.

5. –*Ontologías e Inteligencia Artificial para la Recuperación Eficiente del Conocimiento* [5]. En este articulo muestra la eficiencia del conocimiento eficiente, pero no la eficiencia de la ontología.

4. Justificación

Lo que se propone, es un tipo asistente que servirá para evaluar la coherencia lógica y terminológica de un conjunto de ontologías que contienen una gran cantidad de información relacionada entre sí.

Este asistente será útil para las personas que trabajan con ontologías o pretenden trabajar con ellas, que sirven para crear una base de conocimientos, ya que las ontologías son la base de la web semántica. Con esta herramienta se pretenden disminuir los problemas de ambigüedad de los conceptos que forman parte de los campos semánticos que conforman al dominio de la ontología.

5. Objetivos

5.1 Objetivo general

Diseñar e implementar una herramienta que realice la evaluación automática de la eficiencia lógica y la coherencia terminológica de ontologías públicas.

5.2 Objetivos específicos

- Diseñar e implementar un módulo que permita evaluar la eficiencia lógica de una ontología, calculando el número de iteraciones por segundo requeridas para responder una pregunta.
- Diseñar e implementar un modulo que permita evaluar la coherencia terminológica de una ontología y que muestre el resultado de la evaluación.
- Diseñar e implementar la aplicación donde se integraran los dos módulos de información.

6. Marco teórico

6.1 Ontología

Hace referencia a la formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios dados; con la finalidad de facilitar la comunicación y el intercambio de información entre diferentes sistemas y entidades. Aunque toma su nombre por analogía, ésta es la diferencia con el punto de vista filosófico de la palabra ontología.

Un uso común tecnológico actual del concepto de ontología, en este sentido semántico, lo encontramos en la inteligencia artificial y la representación del conocimiento. En algunas aplicaciones, se combinan varios esquemas en una estructura de facto completa de datos, que contiene todas las entidades relevantes y sus relaciones dentro del dominio.

6.2 Lenguaje OWL

OWL no es el único lenguaje ontológico existente. Previamente han existido un número de propuestas con diversa aceptación. Entre éstas son destacables KIF (Knowledge Interchange Format), DAML+OIL, al que se puede considerar precursor de OWL. Sin embargo, OWL es la propuesta de lenguaje ontológico del W3C, con lo que se ha convertido en el estándar de facto para el desarrollo de ontologías y el punto de referencia para todos los que de un modo u otro, ya sea como usuarios o como desarrolladores de tecnología, trabajan con ellas. La sintaxis de OWL está basada en XML (eXtensible Markup Language), lo que también es cierto del resto de las tecnologías de la Web Semántica. Por utilizar un símil lingüístico, XML es algo así como el alfabeto de la Web Semántica, mientras que RDF (Resource Description Format) aporta el léxico (sobre el que OWL construye significados más complejos).

7. Diseño del sistema

El sistema se conforma por 3 módulos que son los principales como se muestra en la siguiente figura los cuales aseguran la funcionalidad del propósito del sistema.

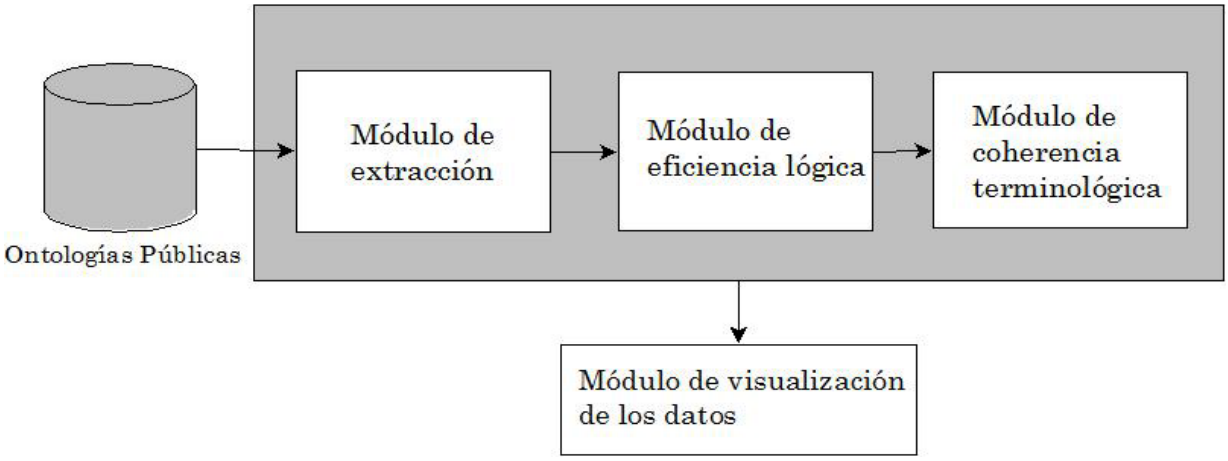

Figura 1. Esquema General del Proyecto.

- Módulo de Extracción: Permitirá ir a buscar las ontologías solo en formato OWL que tienen cierta características que es por ello que el proyecto se enfoco a este lenguaje. Hay que especificar que si se piensa cargar el programa con alguna ontología que no sea OWL no podrá realizarse la ejecución.

  Una carpeta únicamente debe contener los archivos con la extensión OWL la aplicación en la Web debe abrir un directorio local o en su defecto bien podemos pegar la dirección en la cual se encuentra el conjunto de ontologías a analizar en el parseo de sus propiedades.

- Módulo de la eficiencia lógica: detecta durante un tiempo determinado de acuerdo al tamaño de esta el análisis de sus propiedades con la ayuda del api que es el pellet.
- Módulo de coherencia terminológica: esta la podremos visualizar cuando el tiempo estimado sea idóneo de acuerdo a sus propiedades de no ser así la aplicación web nos despliega un comentario detectando alguna inconsistencia después del análisis.
- La visualización de los datos la mandara por medio de un archivo PDF o en su defecto Excel esto con la finalidad de llevar un control de las ontologías con las cuales se están trabajando.

7.1 Proceso de la metodología

Para entender mejor lo que este proyecto realiza, podemos observar la siguiente figura de cómo cada módulo interactúa entre sí.

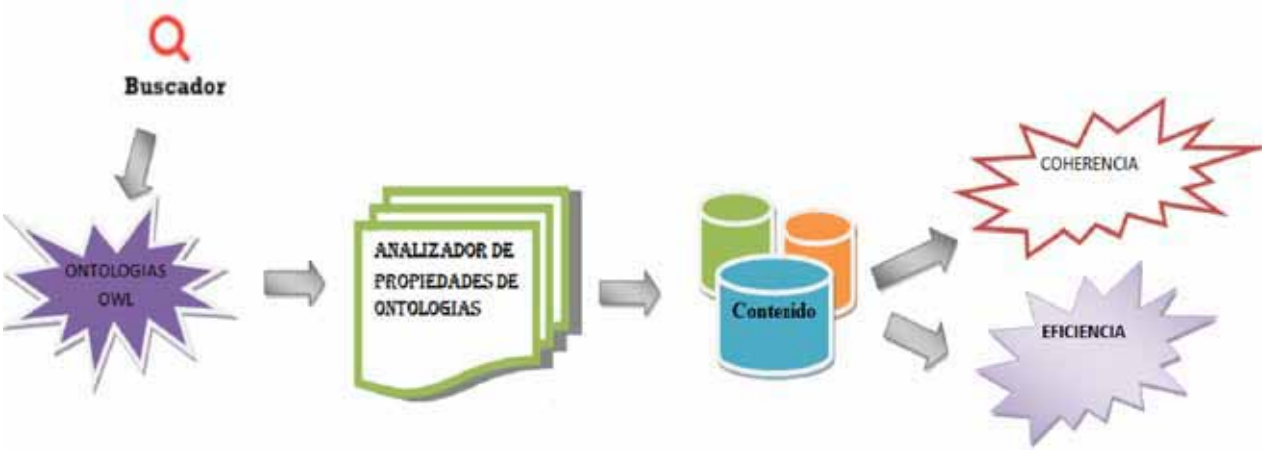

Figura 2. Proceso de análisis de ontologías.

El contenido de la cada una de las ontologías que entran a la acción de poblar en el sistema de identificador de propiedades las cuales muestra el número total de cada una así como todo su contenido desglosado. Algunas de las cuales que trabajamos son las siguientes:

Se incluyen las siguientes características de OWL Lite relacionadas con el esquema RDF.

7.1.1 Class

Una clase define un grupo de individuos que pertenecen a la misma porque comparten algunas propiedades. Por ejemplo, Deborah y Frank son miembros de la clase Persona. Las clases pueden organizarse en una jerarquía de especialización usando subClassOf. Se puede encontrar una clase general llamada Thing que es una clase de todos los individuos y es una superclase de todas las clases de OWL. También se puede encontrar una clase general

llamada Nothing que es la clase que no tiene instancias y es una subclase de todas las clases de OWL.

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</ Owl: ObjectProperty>

<owl:ObjectProperty rdf:ID="course">
  <rdfs:domain rdf:resource="#Meal" />
  <rdfs:range rdf:resource="#MealCourse" />
</ Owl: ObjectProperty>
```

Figura 3. Código de clases de ontologías.

7.1.2 SubClassOf

Las jerarquías de clase deben crearse haciendo una o más indicaciones de que una clase es subclase de otra. Por ejemplo, la clase Persona podría estar definida como subclase de la clase Mamífero. De esto podemos deducir que si un individuo es una Persona, entonces, también es un Mamífero.

7.1.3 Property

Las propiedades pueden utilizarse para establecer relaciones entre individuos o de individuos a valores de datos. Ejemplos de propiedades son tieneHijo, tieneFamiliar, tieneHermano, y tieneEdad. Los tres primeros pueden utilizarse para relacionar una instancia de la clase Persona con otra instancia de la clase Persona (siendo casos de ObjectProperty), y el último (tieneEdad) puede ser usado para relacionar una instancia de la clase Persona con una instancia del tipo de datos Entero (siendo un caso de DatatypeProperty). Ambas, owl:ObjectProperty y owl:DatatypeProperty, son subclases de la clase de RDF rdf:Property.

7.1.4 SubPropertyOf

Las jerarquías de propiedades pueden crearse haciendo una o más indicaciones de que una propiedad es a su vez subpropiedad de una o más propiedades. Por ejemplo, tieneHermano puede ser una subpropiedad de tieneFamiliar. De esta forma, un razonador puede deducir que si un individuo está relacionado con otro por la propiedad tieneHermano, entonces está también relacionado con ese otro por la propiedad tieneFamiliar.

7.1.5 Individual

Los individuos son instancias de clases, y las propiedades pueden ser usadas para relacionar un individuo con otro. Por ejemplo, un individuo llamado Deborah puede ser descrito como

una instancia de la clase Persona y la propiedad tieneEmpleador puede ser usada para relacionar el individuo Deborah con el individuo UniversidadDeStanford.

Cuando definimos una propiedad hay una serie de formas de restringir la relación. El dominio y el rango pueden ser especificados. La propiedad se puede definir como una especialización (subpropiedad) de una propiedad existente. Restricciones más elaboradas son posibles y se describen más adelante.

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</ Owl: ObjectProperty>

<owl:ObjectProperty rdf:ID="course">
  <rdfs:domain rdf:resource="#Meal" />
  <rdfs:range rdf:resource="#MealCourse" />
</ Owl: ObjectProperty>
```

Figura 4. Código para propiedades de individuo.

En OWL, una secuencia de elementos sin un operador explícita representa una conjunción implícita. La propiedad madeFromGrape tiene un dominio de vino y una gama de Winegrape. Es decir, se refiere instancias de la clase Vino a instancias de la clase Winegrape. Varios dominios significan que el dominio de la propiedad es la intersección de las clases identificadas (y de manera similar para la gama).

Del mismo modo, la propiedad supuesto ata una comida a un MealCourse.

Tenga en cuenta que el uso de la gama y la información de dominio en OWL es diferente de la información de tipos en un lenguaje de programación. Entre otras cosas, se utilizan tipos de comprobar la coherencia de un lenguaje de programación. En OWL, una gama puede utilizarse para inferir un tipo. Por ejemplo, dado:

```
<owl:Thing rdf:ID="LindemansBin65Chardonnay">
  <madeFromGrape rdf:resource="#ChardonnayGrape" />
</ Owl: Cosa>
```

Figura 5. Código de coherencia.

8. Desarrollo del Sistema

Una vez que se concluyó la implementación del sistema, fue necesario elaborar pruebas para asegurarse del correcto funcionamiento de todos los módulos. Se realizaron pruebas con diferentes tipos de tamaños en .owl bajo las mismas condiciones de ejecución sin importar el dominio de estas ontologías.

8.1.1 Pruebas para el rastreo de ontologías

En la siguiente figura se muestra como se cargan e insertan las ontologías solo en formato .owl con un tamaño no específico.

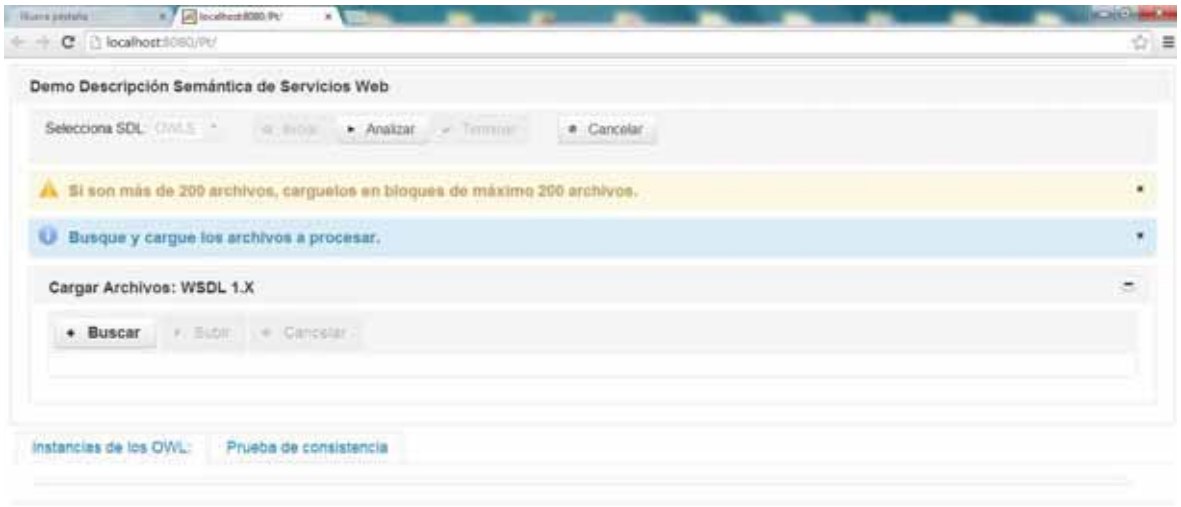

Figura 6. Vista general del proyecto.

El diseño se delimita para hacer una carga máxima de 200 archivos para tener un mejor control de los resultados de la gama de N cantidad de ontologías que se estén trabajando.

8.1.2 Carga de Ontologías.

El módulo de extracción de ontologías hará la función de solo cargar aquellas en el sistema que cumplan el formato, o bien podemos igual localizarlas mediante la dirección de localización.

Puede cargar como ya vimos solo las que estén de forma local no podemos cargar otras que no estén almacenadas posteriormente.

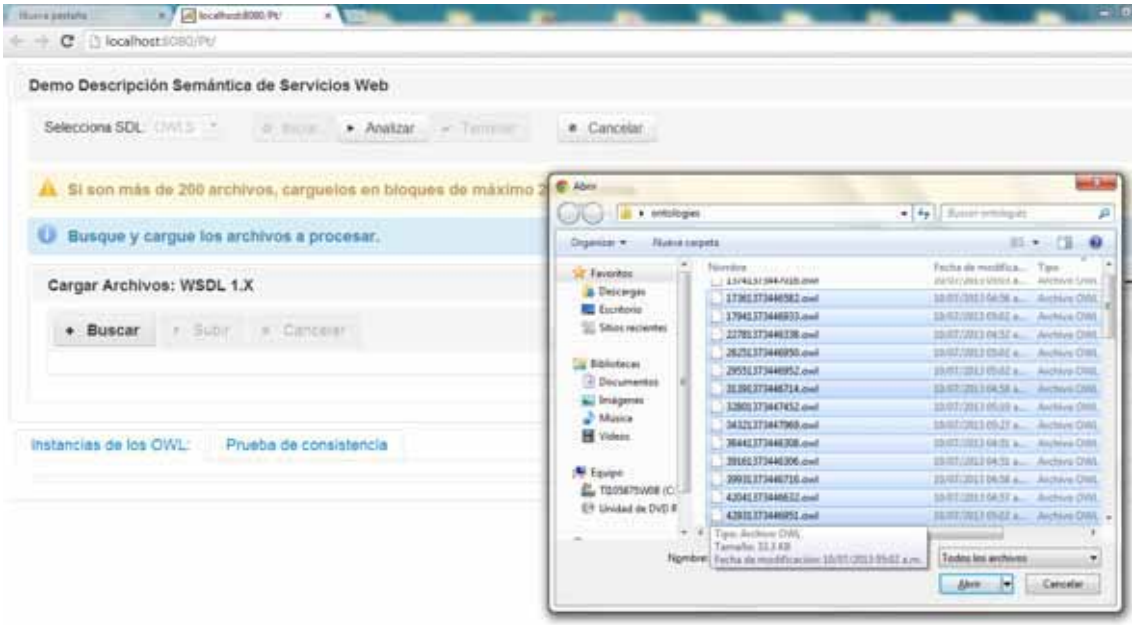

Figura 7. Carga de ontologias.

8.1.3 Población de ontologías.

La localización y la validación de tamaño así como el formato que se tiene contemplado aquí será validado.

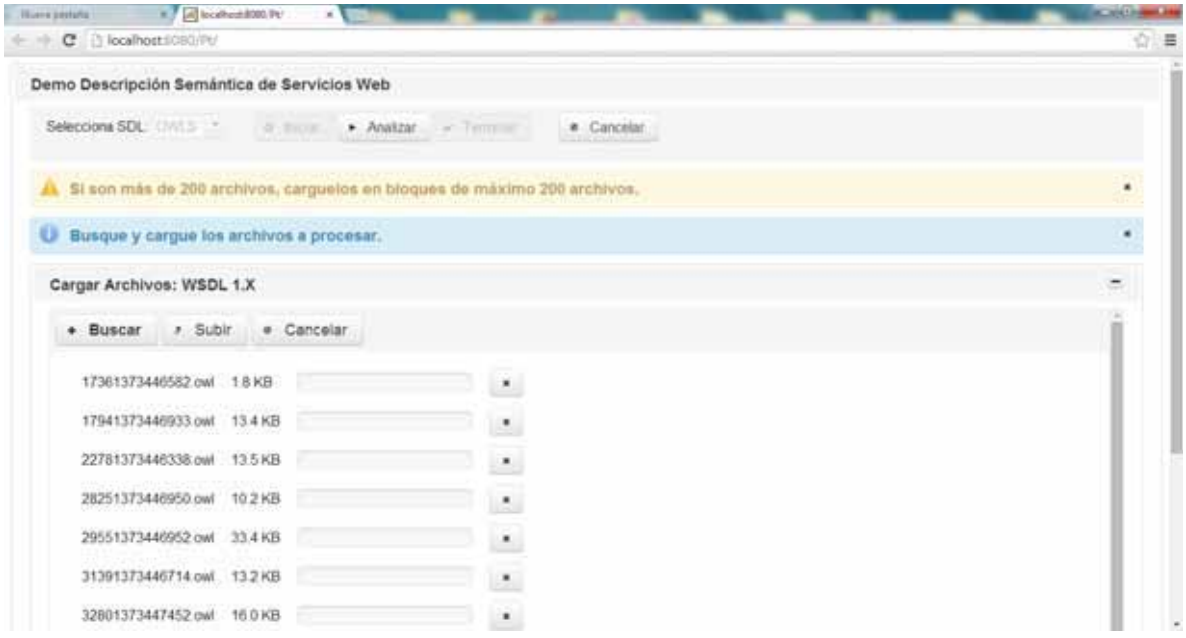

Figura 8. Subir al servicio WEB.

8.1.4 Analizar las ontologías OWL.

En esta parte entra en ejecución el modulo terminología para sacar las propiedades de cada una de las ontologías.

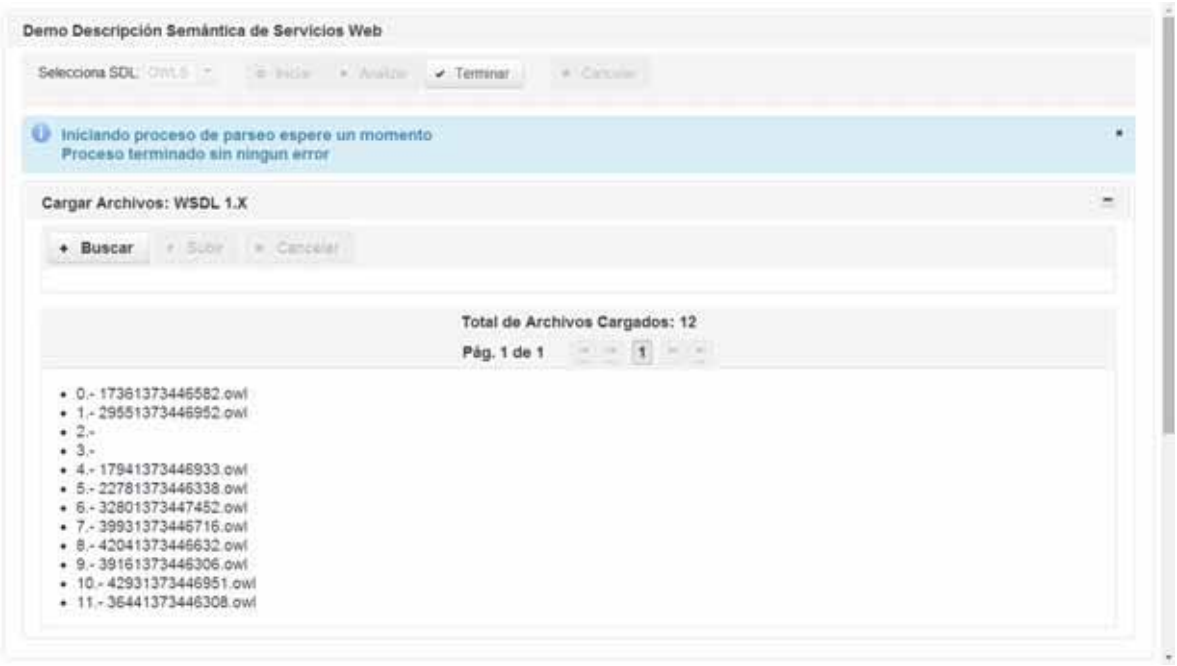

Figura 9. Analisis de OWL.

8.1.5 Módulo de la visualización de los datos

El trabajar con los PrimeFaces da unas platillas para poder apreciar de una forma óptima los datos, para este proyecto la vista fue en forma de cascada mostrando por atributos de cada una de las ontologías.

Figura 10. Parseo de las ontologias.

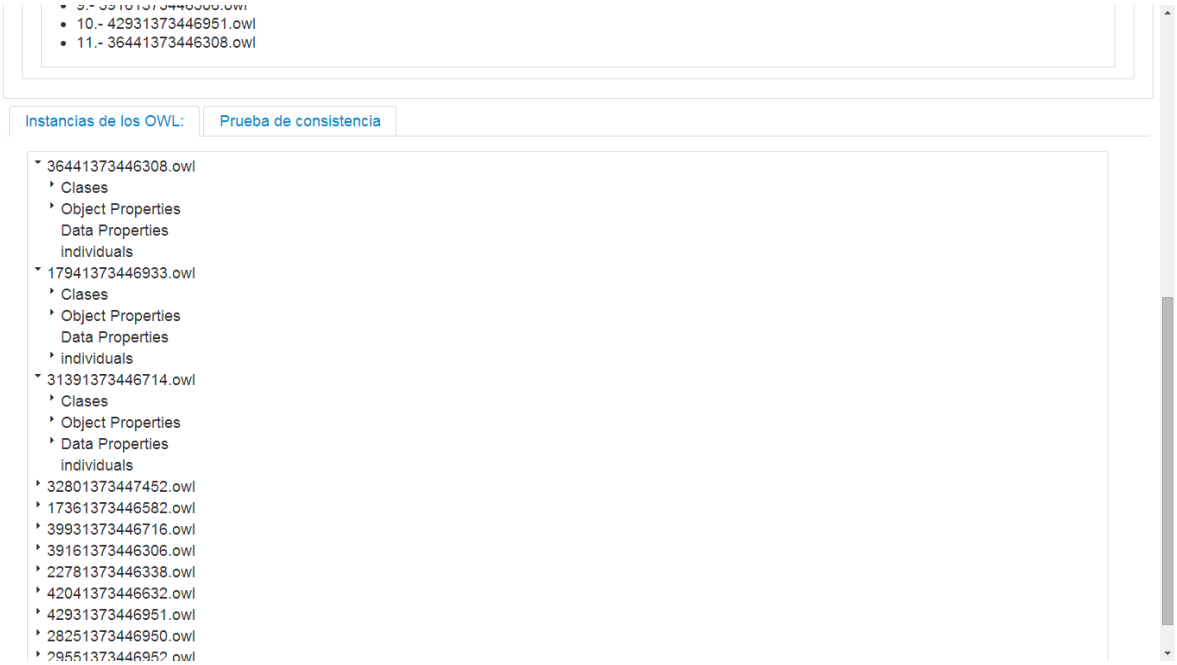

Figura 11. Instancias de los OWL.

Figura 12. Cascada de Instancias OWL.

8.1.6 Prueba de consistencia

Al terminar de examinar las propiedades de los OWL, el módulo de eficiencia lógica junto con la api PELLET nos ayudara al tiempo de acuerdo a su tamaño es aquí donde se dictamina si es eficiente.

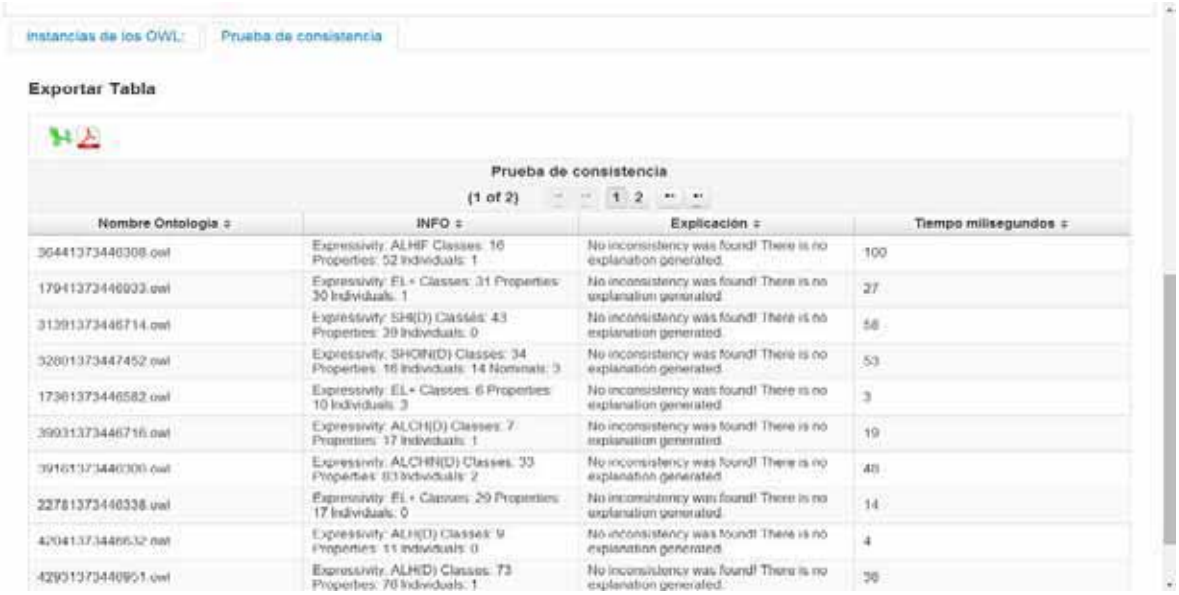

Figura 13. Eficiencia de los OWL.

8.1.7 Manipulación de datos.

Los resultados obtenidos de la coherencia terminológica y la coherencia lógica se manda a exportar para poder manipularlos de acuerdo al objetivo que se tenga.

Figura 14. Exportacion de datos.

9. Pruebas y resultados

Para llevar a cabo este proyecto se conto con lo siguiente:

9.1.1 Recursos

El hardware que se utilizará es una computadora portátil con las siguientes características:

- 600gb de disco duro.
- 4gb en RAM.
- Procesador Intel Core i5 @ 2.67GHz.
- Sistema Operativo W7

El software con licencia libre que se utilizara para el diseño e implementación es el siguiente:

- IDE: NetBeans
- Protégé

- Apache Tomcat
- MySQL Community Server
- API's: OWL-API, Protégé-API, PELLET.

10. Conclusiones

Los lenguajes que se manejan en las ontologias son muy amplios y de los cuales se pueden manipular de una manera en especifica, de acuerdo a los requerimientos de las necesidades de quien trabaje con Ontologias puesto que la Web Semantic es un campo de estudio bastante extenso, y el simple hecho de comprender los comportamientos resulta un reto.

Cuando se anlaizo este proyecto se tuvo que delimitar por lo ya antes mencionado puesto que si bien trabajar con una gama alta de ontologias resulta complicado y saber el comportamiento de ellas en un distinto lenguaje. El proyecto puede crecer por distintas ramas esta herramienta que se alcanzo el solo optimizar la informacion propiedades y tiempos, para aquellos que trabajan con distintas ontologias y saber cuales usar para sus investigaciones o resultados deseados, y poder trabajar con un dominio no especificado, es decir, trabajar con ontologias expresiones ALHI (D), ALH(D), SHI(D), SHOIN (D), por mencionar algunas, personal experto que trabaje con ellas le seran de gran ayuda pues el conocimiento de estas se van coprendiendo mejor y sacarle un mayor provecho de las ontologias que se trabajen con este proyecto.

Los servicios web son muy robustos y son una gran herramienta, y sin duda este proyecto puede ayudar a sacar el maximo rendimiento de un leguaje de ontologias, posteriormente hacer una aplicación movil para un analisis de ontoliogias.

11. Tecnologías de implementación

11.1 NetBeans IDE 7.2.1

Se usara el lenguaje de programación Java. Para tal fin, se empleara este paquete de desarrollo 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 extender el IDE NetBeans.

11.2 Owl

El Lenguaje de Ontologías Web (OWL) está diseñado para ser usado en aplicaciones que necesitan procesar el contenido de la información en lugar de únicamente representar información para los humanos. OWL facilita un mejor mecanismo de interpretabilidad de contenido Web que los mecanismos admitidos por XML, RDF, y esquema RDF (RDF-S) proporcionando vocabulario adicional junto con una semántica formal. OWL tiene tres sublenguajes, con un nivel de expresividad creciente: OWL Lite, OWL DL, y OWL Full.

11.3 Apache Tomcat v 7.0.34

Apache Tomcat es una implementación de software de código abierto que funciona con un contenedor de aplicaciones de Java Servlet y tecnologías JavaServer Pages, se usa en la capa del servidor proporcionando soporte para el servicio web

11.4 OWL API

La API de OWL es una API Java y implementación de referencia para la creación, manipulación y serialising OWL ontologías. La última versión de la API está enfocada hacia OWL 2. Para obtener las últimas actualizaciones, el código y la documentación, visite el nuevo GitHub sitio web.

La API de OWL es de código abierto y está disponible ya sea en virtud de las Licencias LGPL o Apache.

La API de OWL incluye los siguientes componentes:

- Una API para OWL 2 y la aplicación eficiente de referencia en memoria
- Analizador RDF / XML y escritor
- Analizador de OWL / XML y escritor
- OWL analizador de sintaxis funcional y escritor
- Analizador Tortuga y escritor
- KRSS analizador
- OBO formato analizador de archivos planos
- Interfaces de Reasoner para trabajar con los razonadores como hecho + +, ermitaño, Pellet y Racer

11.5 PrimeFaces versión 5

Es una librería de componentes para JavaServer Faces (JSF) de código abierto que cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web. Primefaces está bajo la licencia de Apache License V2.

11.6 Pellet 2.3.1

Implementa un mecanismo denominado E-connect para realizar conexiones a múltiples Ontologías y poder razonar sobre múltiples Ontologías.

12. Bibliografía

[1] T. R. Gruber, “Toward Principles for the Desing of Ontoligies Used for Knowleghe Sharing”, International Journal of Human an Computer Studies, Vol. 43, November, 1995, app. 199-928.

[2] M.J. Sierra, “Sistema clasificador de ontologías mediante métodos de máquinas de soporte vectorial”, Proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México 2013.

[3] L. Y. García, “Sistema de Recuperación de Información de Textos de Investigación de la Web”, Proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México 2013.

[4] A. G. Quezada, “Sistema semántico para la representación de contextos utilizados en aplicaciones de computo obicuo”, Proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México 2012.

[5] J. B. Moreno, “Mapeo de Ontologías Orientado a la Eficiencia”, Articulo de la Universidad de Granada, Departamento de Ciencias de la Computación e Indigencia Artificial.