UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Análisis comparativo de técnicas de minería de datos para el enriquecimiento de estructuras semánticas.

TRABAJO DE TITULACIÓN.

AUTOR: Carrión Pardo, Stalin Samuel.

DIRECTORA: Mora Arciniegas, María Belén**.**

LOJA - ECUADOR

2018

2018

APROBACIÓN DE LA DIRECTORA DEL TRABAJO DE TITULACIÓN

Magister.

María Belén Mora Arciniegas.

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: **Análisis comparativo de técnicas de minería de datos para el enriquecimiento de estructuras semánticas** realizado por **Carrión Pardo Stalin Samuel,** ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, junio de 2018

f). . . . . .

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Yo **Carrión Pardo Stalin Samuel** declaro ser autor (a) del presente trabajo de titulación: Análisis comparativo de técnicas de minería de datos para el enriquecimiento de estructuras semánticas, de la Titulación Sistemas Informáticos y Computación, siendo María Belén Mora Arciniegas director (a) del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f.....
Autor Carrión Pardo Stalin Samuel
Cédula **1104718786**

DEDICATORIA

Primeramente dedico este trabajo a Dios por ser mi objetivo principal en cada momento de mi vida, quien me cuida y vela por mí dándome todo su infinito amor y sabiduría, que me sirve de fortaleza para poder cumplir siempre todos mis propósitos y lograr salir adelante.

A mis queridos padres y hermanas quienes son mi apoyo y fortaleza en el transcurso de mi educación, por siempre brindarme sus consejos y ser mi compañía en cada momento, ayudándome a superar cada inconveniente, fortaleciéndome con su apoyo, sus ánimos y su total confianza para superar cada obstáculo en mi vida y ser una gran persona.

AGRADECIMIENTO

Mi eterno agradecimiento es para Dios quien con su sabiduría, me ha dado la capacidad e inteligencia para poder culminar mis estudios, ayudándome a superar toda adversidad presentada en lo largo de este proyecto, guiándome siempre por el camino del bien. Lo amo infinitamente.

Con mucho amor y respeto agradezco a mis padres quienes son el factor principal durante todo el trayecto de mi formación profesional, brindándome todo su esfuerzo y recursos necesarios para poder lograr mis estudios. Gracias a sus consejos, apoyo y absoluta confianza que me han sabido guiar por el camino del bien, demostrándome sus valores, logrando hacer de mí una persona de bien y perseverancia para cosechar grandes triunfos.

Agradezco a mis queridas hermanas quienes con su apoyo y fortaleza me han dado las fuerzas necesarias para estudiar y lograr cumplir mis objetivos.

Con todo mi amor agradezco a Erika Montero, quien ha estado junto a mí y ha sido mi apoyo en cada momento, gracias por sus ánimos, consejos e infinito amor que me ha sabido demostrar siempre, gracias por ser lo más bonito en mi vida.

Un infinito agradecimiento a mi directora de tesis Magister María Belén Mora Arciniegas, quien ha sido mi guía fundamental durante el trayecto de este trabajo de titulación, gracias por su paciencia, comprensión y por compartir todo su conocimiento, que ha sido el motor esencial para lograr culminar con éxito este proyecto.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE ECUACIONES

RESUMEN

Este trabajo presenta el análisis comparativo de las técnicas de minería ontológica, realizando la descripción del proceso de los algoritmos.

Actualmente la minería ontológica se realiza de forma semi automática y por separado, dejando por fuera todo el conocimiento, obteniendo estructuras semánticas débiles. En donde se implementa un análisis comparativo sobre las técnicas de alineación, enlazado y mezclado de ontologías con el objetivo de integrar las mismas, como primer paso se realiza la alineación, luego el enlazado y mezclado de ontologías. Se efectúa el análisis comparativo entre dos estructuras semánticas, implementando 3 algoritmos de minería ontológica que solventan el enriquecimiento semántico, la alineación utiliza el cálculo de valores de similitudes semánticas, obteniendo las correspondencias finales fuertes, que toman el nombre de alineación de ontologías, el enlazado se realiza en base a la alineación, representando todos los conceptos equivalentes, sin lograr incorporar todo el conocimiento; el mezclado también trabaja sobre la alineación, logrando agregar todo el conocimiento dejado por fuera en el enlazado. Los algoritmos fueron probados en una aplicación web logrando excelentes resultados, pudiendo ser comparados y analizados entre ellos.

PALABRAS CLAVES: Web semántica, minería de datos, minería ontológica, alineación de ontologías, enlazado de ontologías, mezclado de ontologías.

ABSTRACT

This paper presents the comparative analysis of ontological mining techniques, describing the process of the algorithms.

In the first place, the ontological mining is carried out semi-automatically and separately by leaving out all knowledge and obtaining weak semantic structures. Then, the comparative analysis is implemented on the techniques of alignment, linking and mixing of ontologies in order to integrate them. The first step is the alignment, then the linking and mixing of ontologies. The comparative analysis between two semantic structures is carried out into 3 ontological mining algorithms. First is to solve the semantic enrichment, then the alignment uses calculation of values of semantic similarities to obtain the strong final correspondences which take the name of ontology alignment. The linked it is done based on the alignment which represents all the equivalent concepts without being able to incorporate all the knowledge. The mixing also works on the alignment and managing in order to add all the knowledge left out in the link. The algorithms were tested in a web application, which helped achieved excellent results of being able to be compared and analyzed among them.

KEYWORDS: Semantic Web, data mining, ontological mining, ontology alignment, linking of ontologies, mixed ontologies.

CAPITULO I INTRODUCCION

1.1 Introducción

En los últimos años se ha venido desarrollando la minería ontológica de manera automática o semi automática, realizando por separado el proceso de alineación, enlazado y mezclado de ontologías, dejando por fuera todo el conocimiento al momento de realizar la unión de las mismas. Por lo que nace la necesidad de aplicar la minería ontológica, con el objetivo de abarcar todo el conocimiento que es dejado fuera, trabajando de manera conjunta y no por separado logrando obtener un total enriquecimiento semántico, las técnicas de minería ontológica al ser analizadas y procesadas de manera considerada pueden incluir todo el conocimiento y contribuir con una gran ventaja para las estructuras semánticas. En la actualidad se pretende mejorar el enriquecimiento semántico con técnicas que permitan incluir de manera completa todo el conocimiento que no es tomado en cuenta, para esto se considera ontologías que pertenezcan al mismo dominio y que pretendan ser enriquecidas.

Los procesos de la minería ontológica ayudan a solventar problemas al momento de aplicar la unión de las mismas, debido a que se han convertido en metodologías fuertes y estructuradas para el procesamiento de datos semánticos, a través de una serie de algoritmos que permiten agrupar ontologías del mismo dominio en un solo resultado, enriqueciendo las mismas; además se utilizan herramientas de computación como: Protegé, y lenguajes de programación que ayudan a trabajar en el correcto proceso de la información.

En el presente trabajo la problemática es solucionada a través de las técnicas de minería de datos que permiten el enriquecimiento del conocimiento semántico. Para alcanzar el objetivo del proyecto, se desarrolla en orden secuencial la alineación, enlazado y mezclado que permiten integrar las estructuras semánticas, en caso de que las ontologías no hayan cumplido el primer paso de alineación no se puede proceder a enlazar o mezclar las mismas.

Para aplicar la minería ontológica se analiza varias ontologías que sea de un mismo dominio, se estudia y determina ciertos algoritmos para las 3 técnicas, en el caso de la alineación: KNN-Vecinos y AdaBoost, enlazado general para el enlazado de ontologías, mezcla fuerte y débil para el mezclado; que logran asociar varias matrices de medidas de similitudes para la correspondencia fuerte semántica. Estas técnicas permiten analizar de manera correcta todos los conceptos integrados, es decir cuál de estas agrega todo el conocimiento o deja el mismo por fuera.

1.2 Planteamiento del problema

El proceso de alineación, enlazado y mezclado de ontologías se desarrolla por separado, esto debido a que actualmente no existe alguna aplicación que logre realizar el enriquecimiento de estructuras semánticas de manera conjunta; por ejemplo, no existe ningún desarrollo sobre enlazado de ontologías que pretendan unir las mismas, pero si se han desarrollado mezcladores semi automáticos con el desperfecto de no lograr incluir todo el conocimiento, es decir obteniendo resultados incompletos.

En el campo de la Web Semántica y minería de datos se dispone de una enorme cantidad de datos que no se encuentran integrados, pero si realizamos una integración de alineación, enlazado o mezclado de manera separada, estaremos dejando por fuera todo el conocimiento, que presentaría como resultado una ontología demasiado pequeña y débil; esto puede ocurrir al momento de aplicar la alineación, en donde no se puedan encontrar todas las similitudes semánticas, obteniendo bajas correspondencias en relación a la cantidad de datos pertenecientes a las ontologías a ser procesadas. Lo mismo sucede al momento de aplicar un enlazado o mezclado por separado, se obtiene un enriquecimiento demasiado básico o débil.

Por lo que en el presente trabajo se permite tener todo el conocimiento unificado en una sola ontología, aplicando todas las técnicas de minería ontológica, de una forma conjunta y secuencial, con el propósito de obtener un completo enriquecimiento semántico.

1.3 Objetivos

1.3.1 Objetivo general.

Analizar diferentes técnicas de minería de datos para el enriquecimiento de recursos semánticos.

1.3.2 Objetivos específicos.

- Investigar las técnicas de minería de datos para el enriquecimiento de estructuras semánticas (ontologías).
- Analizar comparativamente técnicas de minería de datos que permitan enriquecer estructuras o recursos semánticos.
- Definir un conjunto de estructuras semánticas (ontologías) para evaluar el uso de técnicas de minería de datos.
- Implementar una técnica de minería de datos en un conjunto de estructuras semánticas (ontologías).
- Realizar pruebas de los resultados.
- Presentar y validar los resultados obtenidos.

1.4 Metodología de desarrollo

En el presente trabajo de titulación se desarrolla una metodología de investigación cuantitativa, es objetiva y orientada a resultados exactos, permite la toma de decisiones entre varias alternativas durante todo el proceso. Esta investigación aplica métodos estadísticos sobre la cantidad de datos que se estudian (Fernández & Pértegas, 2002).

A continuación, se describen los pasos de la metodología implementada que permiten cumplir con los objetivos de este trabajo.

Primer paso: definir el alcance del proyecto (se detalla cuál es el fin del trabajo de titulación y a donde se logra llegar con el proceso del mismo), se plantea el problema, objetivo general y específicos, que metodología se desarrolla y la organización del trabajo.

Segundo paso: Se desarrolla el capítulo correspondiente al marco teórico definiendo los conceptos y conocimientos sobre Web Semántica, ontologías, minería de datos y minería ontológica.

Tercer paso: se define las técnicas de minería ontológica con sus respectivos ejemplos y algoritmos, sobre la alineación, enlazado y mezclado de ontologías.

Cuarto paso: implementación de las técnicas de minería de datos para el enriquecimiento de estructuras semánticas, seleccionando el dominio y ontologías a trabajar.

Quinto paso: análisis comparativo de los resultados de los algoritmos de minería ontológica sobre los datos enriquecidos.

Sexto paso: desarrollo de trabajos futuros, conclusiones y recomendaciones del presente trabajo de titulación.

1.5 Organización del proyecto.

El presente trabajo de titulación se ha elaborado en seis capítulos con seis anexos, la misma estructura se define a continuación:

Capítulo 1: se define la introducción, planteamiento del problema, objetivos generales y específicos y metodología de desarrollo.

Capítulo 2: desarrollo del marco teórico con las temáticas como: Web Semántica, minería de datos, ontologías, minería ontológica y técnicas de minería ontológica.

Capítulo 3: se desarrolla las técnicas de minería ontológica con ejemplos y algoritmos sobre alineación, enlazado y mezclado de ontologías.

Capítulo 4: se describe la implementación de la minería ontológica, se desarrolla un aplicativo sobre las técnicas de alineación, enlazado y mezclado ontologías.

Capítulo 5: Se obtiene el análisis comparativo de los resultados de cada una de las técnicas de minería ontológica, añadiendo los trabajos futuros, conclusiones y recomendaciones.

CAPITULO II MARCO TEORICO

En el presente capitulo se desarrollan temas sobre la Web semántica, minería de datos, ontologías, minería ontológica con sus respectivas técnicas existentes y trabajos relacionados que ya han aplicado mezclado, enlazado y alineación de las ontologías.

2.1 Web Semántica

Es una Web muy amplia, útil y fácil, en donde cualquier usuario que se encuentre navegando en el Internet puede encontrar respuestas a sus consultas de manera simple y acelerada, obteniendo el procesamiento de información y transferencias de manera eficaz, la Web semántica tiene la capacidad para que las computadoras puedan resolver problemas utilizando operaciones sobre datos definidos, se compone por metalenguajes y estándares de representación como: XML, XML Schema, Ontologías, RDF, RDF Schema, SPARQL y OWL (Zagal, 2008). Hoy en día la Web Semántica tiene una importante etapa en la evolución de las ontologías por lo que la información se encuentra bien definida, lo que implica que exista una mejor cooperación al momento de trabajar entre las personas y computadoras sin importar el lugar donde nos encontremos (Caliusco & Stegmayer, 2010).

Actualmente la Web Semántica ha logrado adquirir una enorme relevancia dentro de las TIC (Tecnologías de información y comunicación), aplicándose de manera principal en la educación, su objetivo principal es que el conocimiento es representado en contenidos de los recursos Web, que sean de manera útil en las aplicaciones, permitiendo que las máquinas entiendan de manera precisa el significado de los datos y documentos, para recoger todo este conocimiento se debe manejar ontologías definiendo los conceptos y relaciones que se presenten en el dominio de la aplicación (Fernández, 2016). A continuación en la figura 1 se indican las principales capas y componentes de la Web Semántica (Berners, 2001):

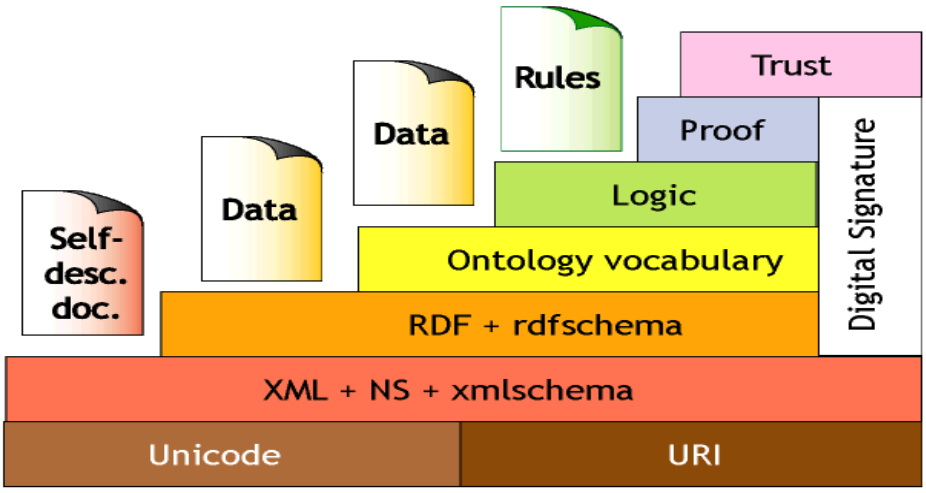

Figura 1: Arquitectura de la Web Semántica
Fuente: (Berners, 2001)
Elaboración: (Berners, 2001)

- **UNICODE:** Permite utilizar la decodificación de texto con caracteres especiales, con el propósito de mostrar la información de la Web Semántica en todos los idiomas (Lamarca, 2013).
- **URI:** Uniform Resource Identifier (identificador de recursos uniforme), permite situar los recursos que son presentados a través del Internet (Lamarca, 2013).
- **XML:** Extensible Markup Languaje (lenguaje de marcado extensible), Metalenguaje que presenta una estructura definida sobre los documentos, pero sin lograr describir su significado (Zagal, 2008).
- **XML Schema:** Esta capa se encarga de unir varias tecnologías, su objetivo es que los agentes se entiendan entre ellos, es una de las capas más técnicas de la Web Semántica. Además define la estructura de los documentos XML (Lamarca, 2013).
- **RDF:** Resource Description Framework (marco de descripción de recursos), es un lenguaje con reglas semánticas que descompone el conocimiento en pequeñas partes, definiendo sujeto (recursos a los cuales se refiere), predicado (los recursos que se definen) y objeto (recursos que se terminan de definir), aportando una semántica para ser representada en XML(Nigro & Císaro, 2006).
- **RDF Schema:** Esta capa se apoya sobre XML Schema, definiendo un lenguaje universal con el objetivo de indicar diferentes ideas sobre la Web Semántica, define un vocabulario basado en RDF. Esta capa ofrece una semántica definida de manera clara entre propiedades y clases (Lamarca, 2013).
- **LENGUAJE DE ONTOLOGIAS:** Capa que permite clasificar la información, extendiendo la representación de la Web Semántica, permite agregar clases y propiedades en los recursos (Lamarca, 2013).
- **LÓGICA:** Son reglas de inferencia, que pueden utilizar las ontologías con el fin de concluir lo que se necesita y entender o tratar los términos de una manera más eficiente (Fernández, Carbonell, Pérez, & Villalón, 2009).
- **FIRMA DIGITAL:** Datos que serán manejados en el computador, a través de esta capa los datos se encriptan de manera segura (Lamarca, 2013).
- **ONTOLOGÍAS:** Permiten representar el conocimiento, definiendo un conjunto de conceptos entre relaciones o reglas con un determinado vocabulario, las ontologías son trabajadas en bases de datos a través de aplicaciones y mediante usuarios, con el propósito de compartir información específica (Nigro & Císaro, 2006).
- **SPARQL:** Lenguaje basado en consultas RDF para recursos semánticos, utilizando diferentes fuentes de datos (Fresno, 2014).

- **OWL:** Este lenguaje presenta una estructura para ontologías mediante un vocabulario, muestra las clases y sus propiedades como las conceptualizaciones y relaciones que existen en sus clases (Gómez, 2013).

2.2 Ontologías

Las ontologías representan un área de conocimiento, son manejadas por personas, aplicaciones y base de datos que comparten un dominio de información; un dominio es un espacio o lugar específico de conocimiento, como medicina, biología, informática, fabricas, automóviles información financiera (Fernández & Carbonell, 2009), etc.

Una ontología es un conjunto de hipótesis explicitas, presenta tipos de propiedades y relaciones entre objetos pertenecientes a un dominio; las ontologías presentan un orden específico, en donde cada vocabulario indica nombres como predicados unarios o binarios conocidos como conceptos y relaciones, describen una jerarquía de conceptos relacionados: cuando existen casos sofisticados, los axiomas adecuados se añaden con el fin de expresar otras relaciones (Caliusco & Stegmayer, 2010). Dos ontologías pueden presentar un vocabulario diferente como por ejemplo: pueden estar constituidas por palabras en inglés y español, pero en si comparten la misma conceptualización. Una conceptualización se define como una estructura (D, R) en donde D es el dominio y R es el conjunto o relaciones que pertenecen al dominio. Las ontologías presentan conjuntos de axiomas lógicos diseñados para tener claro el significado de un vocabulario deseado; los axiomas son diseñados de tal manera que el conjunto de modelos se aproxime a la mejor manera posible del conjunto de los modelos previstos (Guarino, 2006). Para poder definir el vocabulario de una ontología se define las reglas con los términos y las condiciones que se van a combinar (Caliusco & Stegmayer, 2010).

Las ontologías presentan vocabularios comunes, estos vocabularios pueden ser reusables o propios; facilitan la comunicación de los metadatos (datos que describen otros datos), cada vocabulario esta descrito por un documento señalado por una URI, la URI hace referencia a las clases y propiedades del vocabulario; una URI está compuesta desde el encadenamiento del vocabulario con el nombre de la clase o propiedad. Existen cientos de vocabularios, a continuación se nombran algunos de los más reconocidos:

DUBLIN CORE: vocabulario para la recuperación de recursos, contiene términos DCMI.

FOAF: representa relaciones procesadas por máquinas de una manera fácil, a través de XML y RDF.

SCHEMA.ORG: vocabulario con una base de esquemas, trabaja en etiquetado de datos constituidos en páginas web.

SKOS: incorpora el contenido de esquemas que se identifican mediante una URI (Open, Hosted by the Knowledge Foundation, 2016)**.**

2.2.1 Componentes principales de una ontología.

Las ontologías están conformadas o constituidas por clases, en las cuales cada clase derivan propiedades propias de la clase. A continuación en la Figura 2 se presenta un ejemplo sobre ontologías, para representar las características que definen el perfil de un estudiante y una ontología del modelo de un dominio, en donde un agente puede acceder a ambas ontologías para seleccionar y mostrar al alumno los elementos el material educativo (Gascueña & Fernandez, 2005)

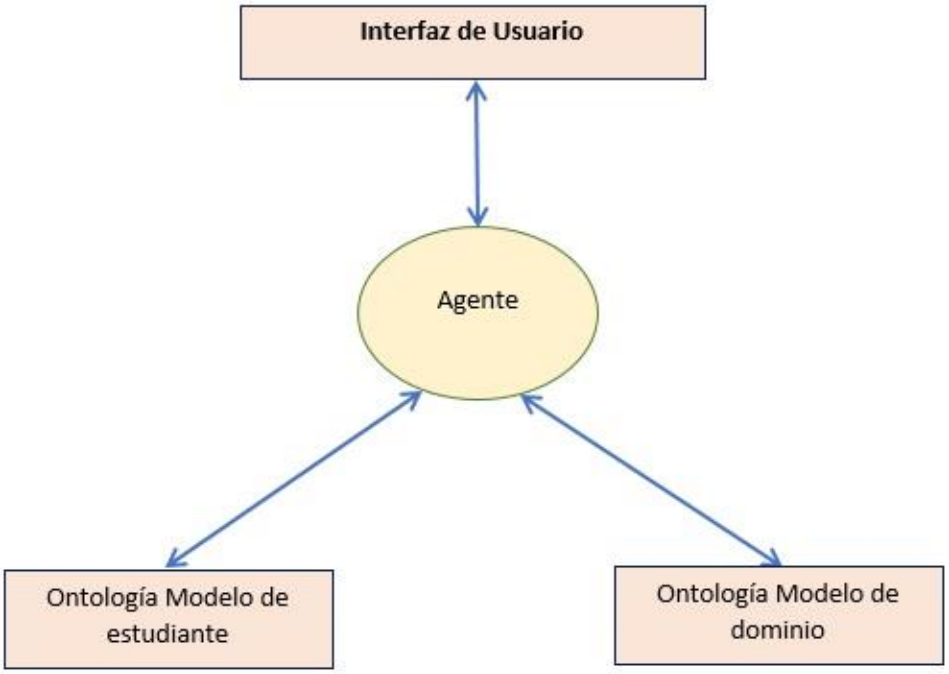

Figura 2: Arquitectura adaptativa al usuario
Fuente: (Gascueña, & Fernandez, 2005)
Elaboración: (Gascueña & Fernandez, 2005)

2.2.2 Ejemplos de Ontologías

Ejemplo de una Ontología modelo alumno.

En la Tabla 1, se indica una ontología que representa todo el conocimiento que presenta el estudiante, en donde la clase principal será denominada estudiante y sus respectivas propiedades. (Gascueña & Fernandez, 2005).

Tabla 1: Ontología que representa el perfil de un alumno

Clase	Propiedades
Estudiante	• Nombre • Apellido • Cédula • Email • Teléfono • Sexo • Año • Domina lenguaje
Estilo de aprendizaje	• Sensorial intuitivo • Visual verbal • Activo reflexivo • Secuencia global
Tiene conexión de red	• Nombre • Velocidad
Tiene preferencias visuales	• Color fondo • Color titulo • Color párrafo
Estilo aprendizaje	• Tiene objetivo aprendizaje • Aprende

Fuente: (Gascueña & Fernandez, 2005)
Elaboración: (Gascueña & Fernandez, 2005)

La Figura 3 presenta el modelo de una ontología del dominio del alumno, con sus respectivas clases, propiedades y atributos.

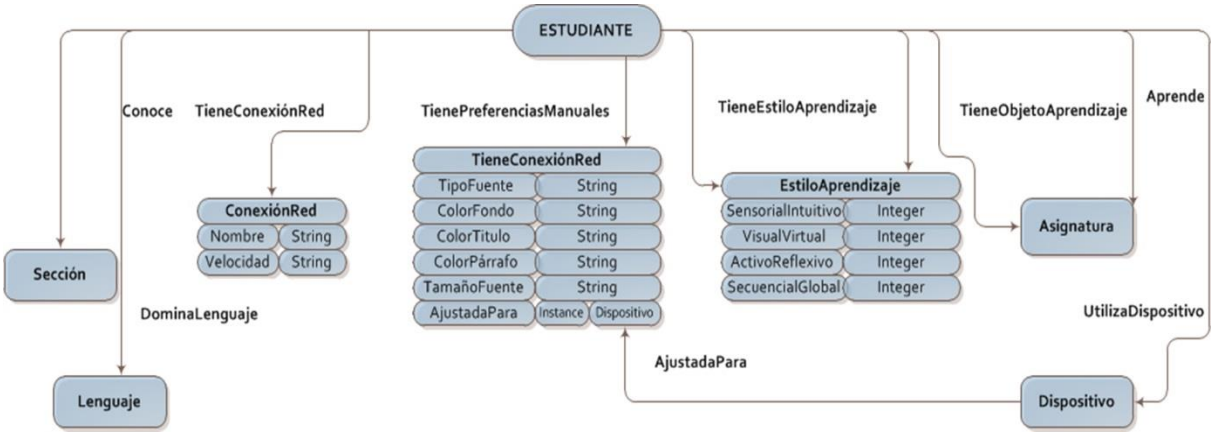

Figura 3: Ontología del modelo estudiante
Fuente: (Gascueña & Fernandez, 2005)
Elaboración: (Gascueña & Fernandez, 2005)

Ejemplo de una Ontología de un dominio universitario

En la Figura 4, se presenta un ejemplo de un dominio universitario, en donde la universidad está estructurada por departamentos, cada departamento tiene un área de conocimiento; cada asignatura es dictada por varios profesores y cada asignatura pertenece a un área de conocimiento departamento (Gascueña & Fernandez, 2005).

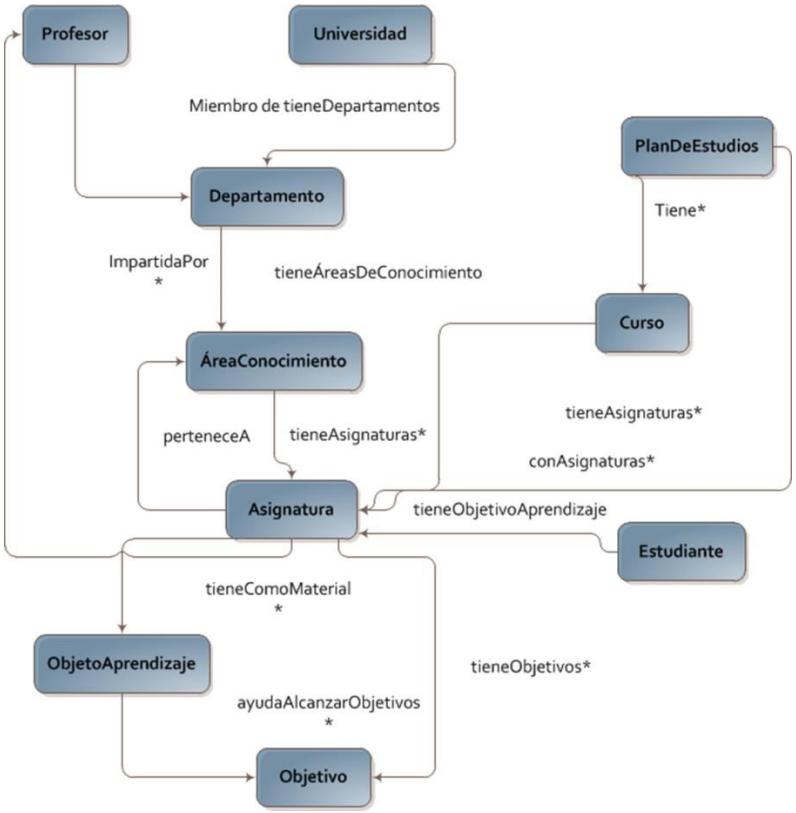

Figura 4: Estructura organizativa
Fuente: (Gascueña & Fernandez, 2005)
Elaboración: (Gascueña & Fernandez, 2005)

2.3 Minería de Datos

Vivimos en una era moderna en donde trabajamos de forma voluntaria o involuntaria, almacenando extensas cantidades de datos, por lo que todos los días se crea un sinnúmero de información, la misma es almacenada de manera automática y se genera cuando realizamos muchas cosas en el Internet, por ejemplo al momento que se utiliza redes sociales, o a su vez se efectúan transacciones interbancarias ya sea con tarjetas de instituciones financieras o simplemente cuando se navegue de manera normal en el Internet, toda esta información se encuentra esparcida en las redes, la información que se forma y almacena depende del tipo de entidad o persona que genere, por ejemplo empresas o entidades muy grandes crean enormes conjuntos de datos e información de acuerdo al servicio que prestan. Con la era moderna las redes sociales cumplen un papel fundamental en las fuentes de datos, son expresadas a través de: videos, imágenes, blogs, etc (Han & Kamber, 2006)

Algunos autores proponen un Concepto de Minería de Datos:

La minería de datos es el proceso que consiste en la extracción del conocimiento de bases de datos; examina y analiza grandes cantidades de datos. La minería de datos posee herramientas para extraer patrones y así permitir una mejor comprensión de los datos para poder realizar predicciones de comportamientos futuros (Pérez & González, 2007).

La minería de datos es un procedimiento que permite descubrir el conocimiento de bases de datos, basándose en algoritmos de análisis de datos, permitiendo encontrar restricciones de eficiencia que provocan una lista de patrones sobre la información. Este método permite elegir, buscar y formar enormes cantidades de datos mediante técnicas de inteligencia artificial, con el objetivo de descubrir patrones o relaciones ocultas (Gonzáles, 2010).

La minería de datos presenta las siguientes etapas:

- **Limpieza de datos:** se encarga de borrar todos los datos que presentan inconsistencias.
- **Integración de datos:** permite integrar o mezclar datos desde diferentes fuentes.
- **Selección de datos:** ayuda a seleccionar y recuperar los datos más importantes de una base de datos para luego ser analizados.
- **Transformación de datos:** esta etapa se encarga de transformar los datos de manera adecuada, para luego ser aplicados a la minería.
- **Minería de datos:** su objetivo principal es utilizar procesos inteligentes para seleccionar patrones de conocimiento.
- **Evaluación de patrones:** esta capa identifica, evalúa y selecciona los patrones con mayor interés.

- **Presentación del conocimiento:** permite mostrar al usuario el conocimiento seleccionado (Valencia, 2008)

2.4 Minería Ontológica

Como se ha venido mencionando las ontologías nos permiten representar el área del conocimiento de una manera extensa, pero actualmente con el enorme crecimiento de las mismas es necesario de unir la minería de datos con las ontologías, creando lo que es la minería ontológica (ontology mining), que representa un conjunto de actividades para el proceso de desarrollo de las estructuras semánticas, con metodologías lenguajes y herramientas que son necesarias para el diseño de ontologías.

Esta área permite obtener esquemas de comportamiento, usando diferentes técnicas de minería ontológica, con el propósito de generar enriquecimiento semántico, reutilizando ontologías de dominio general y patrones de diseño de ontologías, con el propósito de ayudar el modelado, dinámica y evolución de la semántica; explotando con mayor profundidad todas las técnicas que existen para poder extraer el conocimiento global de un conjunto de ontologías, logrando obtener un dominio de conocimiento más amplio, con resultados completos y de fácil entendimiento (Aguilar, 2015).

2.4.1 Técnicas de Minería Ontológica

Actualmente existen diferentes técnicas de minería ontológica que permiten realizar la unión de ontologías sin dejar por fuera el conocimiento, a través de una alineación, enlazado y mezclado, permitiendo analizar la información de manera distinta y eficaz, obteniendo como resultado todo el conocimiento semántico de manera única. Permiten obtener información adicional a un conjunto de ontologías, estas técnicas son: alineación, enlazado y mezclado de ontologías (Rangel, 2015).

2.4.1.1 Alineación de ontologías.

La técnica de alineación de ontologías se da a través del proceso de correspondencias semánticas a través de dos ontologías, permite construir y definir la similitud entre todos los nodos pertenecientes que cumplen el proceso de análisis, obteniendo como resultado las equivalencias entre entidades de las diferentes estructuras semánticas (Zagal, 2008).

2.4.1.2 Enlazado de ontologías.

El enlazado de ontologías es una técnica que realiza un proceso de conectividad entre estructuras semánticas, sin requerir que las mismas sean mezcladas, permite encontrar relaciones entre entidades que son o pertenecen a diferentes ontologías, se elabora un

conjunto de reglas, permitiendo realizar consultas que puedan obtener toda la información de los datos semánticos (Rangel, 2015).

2.4.1.3 Mezclado de ontologías.

El mezclado es otra de las técnicas de la minería ontológica, su proceso consiste en unir varias ontologías entre el mismo dominio para establecer el conocimiento, cuando existe el mezclado, se da en el mismo conocimiento pero con diferentes representaciones y en donde las ontologías pueden o no concordar en sus conceptos, permite tener el conocimiento de manera local, esto es muy útil en los sistemas distribuidos ya que permite consultar en el mismo dominio facilitando la información y evitando consultas en varios sistemas semánticos (Aguilar, 2015).

2.5 Trabajos Relacionados

Existen diferentes investigaciones que han aplicado la alineación, enlazado y mezclado de ontologías, las mismas exponen cómo se han aplicado en cosas de la vida cotidiana, por lo que a continuación se indican estos resultados.

2.5.1Dominio geoespacial: Turismo electrónico

La investigación de Zagal (2008) desde las paginas 64 hasta la 87, alineación de ontologías utilizando el método de Boosting, es un robusto procedimiento que combina varios modelos de datos fuertes, proyectando resultados exactos e individuales, realiza un prototipo de alineación de datos semánticos en el dominio geoespacial, alineando estructuras semánticas con servicios turísticos ubicados en las ciudades de Acapulco y Cancún pertenecientes al país de México, desarrollando un sistema Web OntoMashup que está dirigido a este tipo de lugares, se ubica geográficamente hoteles que pertenecen a las ciudades mencionadas anteriormente, la información de las dos ontologías es trabajada adoptando las utilidades de Google Maps y Geonames.org; los datos semánticos presentan significados de servicios turísticos, utilizando una organización ordenada de conceptos, teniendo como resultado las similitudes globales y locales, la Figura 5 indica el esquema sobre la alineación, que tiene como entrada dos ontologías y la salida es un conjunto de correspondencias.

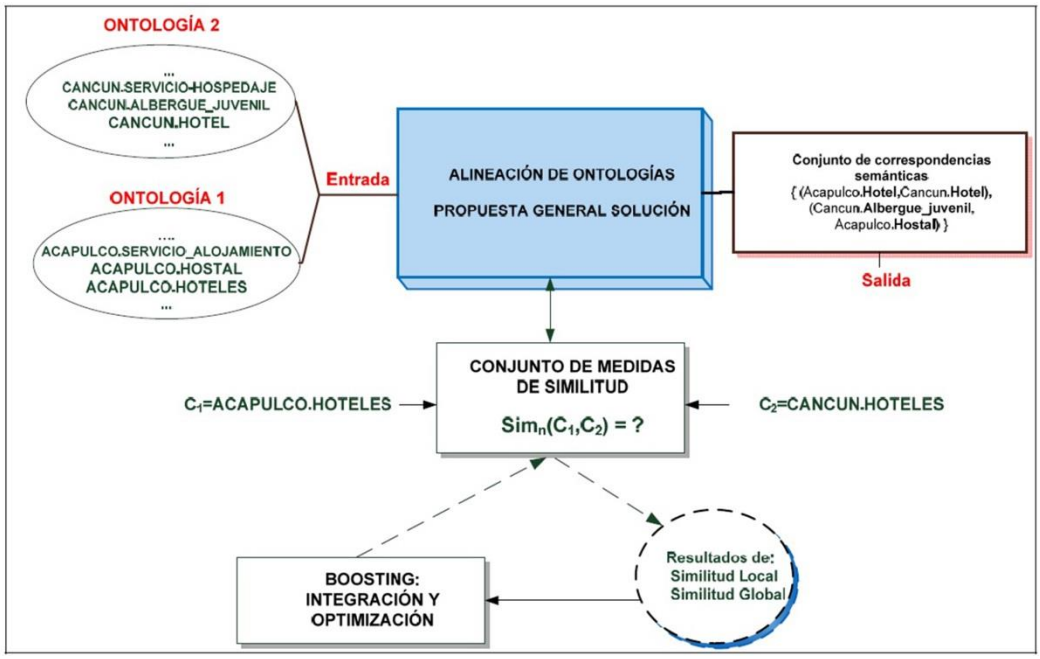

Figura 5: Esquema de alineación de ontologías
Fuente: (Zagal, 2008)
Elaboración: (Zagal, 2008)

La Figura 6, muestra la interfaz gráfica sobre alineación de ontologías para entornos turísticos.

Figura 6: Interfaz de consulta de OntoMashup
Fuente: (Zagal, 2008)
Elaboración: (Zagal, 2008)

La Figura 7, presenta los resultados finales alineados, especificando los nombres de los hoteles.

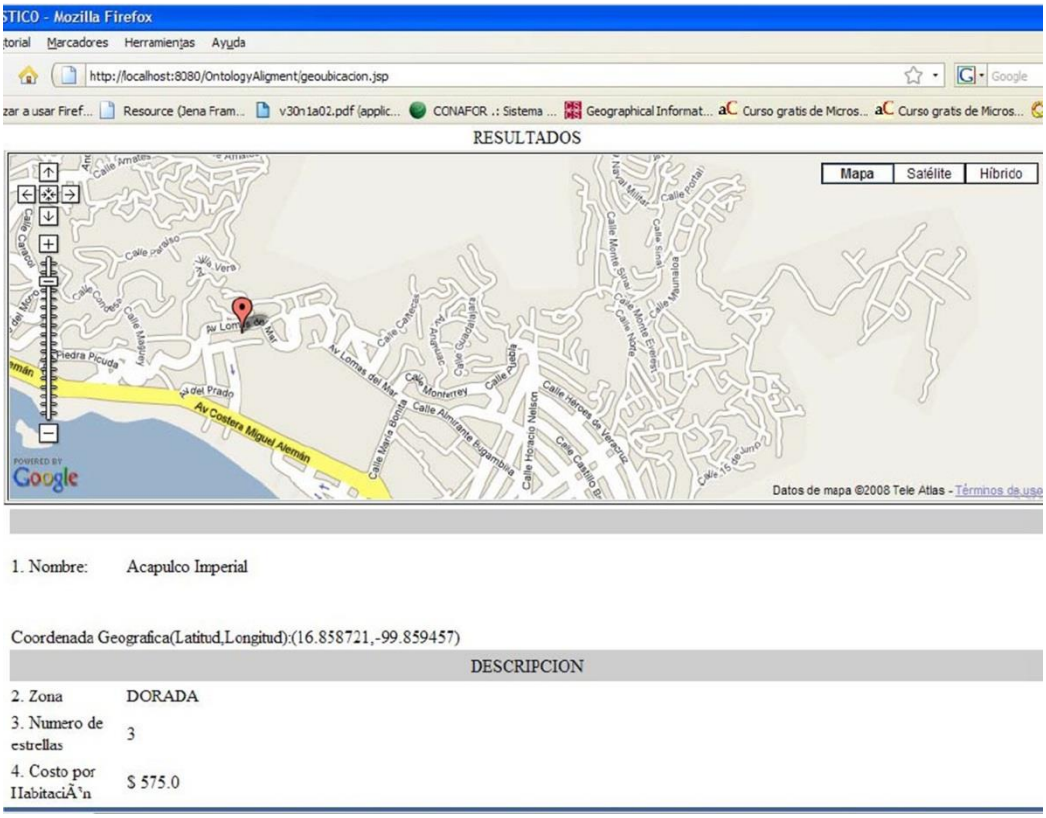

Figura 7: Resultados para la búsqueda por nombre de hotel
Fuente: (Zagal, 2008)
Elaboración: (Zagal, 2008)

2.5.2 OEGMerge: un modelo de mezcla de ontologías basado en casuísticas

Según Fernández, Gómez & Ramos (2005) el modelo OEMERGE nace a partir de dos estructuras semánticas originarias, obteniendo como resultado una tercera ontología, la misma que contiene todo el conocimiento original de manera agrupada. Estos autores han desarrollado un software para la automatización de este proceso, en una plataforma de minería ontológica conocido como WebODE ODEMerge, realiza la mezcla de manera automática entre dos estructuras semánticas, el grupo de minería ontológica de la Universidad Politécnica de Madrid UPM ha realizado varios ejemplos con esta aplicación; para realizar la mezcla ontológica primero se debe realizar el mapping como se indica en la Figura 8 mapping entre ontologías(Chavez, 2006).

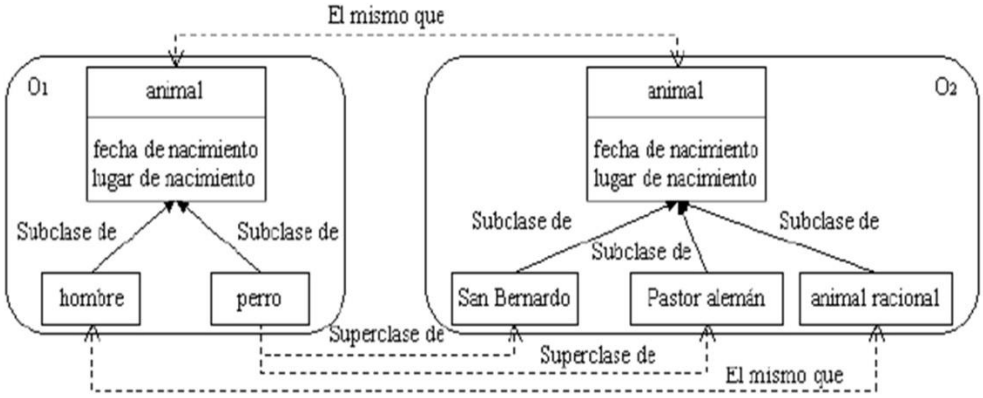

Figura 8: Mapping entre dos ontologías
Fuente: (Chavez, 2006)
Elaboración: (Chavez, 2006)

La Tabla 2, indica ciertas reglas que utiliza OEGMerge para el mezclado de ontologías, en donde si el caso es 1 determinando que ambos conceptos sean iguales se procede a realizar una mezcla identidad, para el caso 2 y 3, cuando uno de los conceptos sea una clase padre o una subclase a mezclarse se realiza una inserción de una subclase.

Tabla 2: Reglas de definición de la función Mezcla Ontología

Caso	Condición	Acción
1	$elMismoQue_M(A_{aux}\ C_{base})$	$MezclaIdentidad(C_{aux}\ C_{base}\ O_{aux}\ O_{base}$ mapping)
2	$superclaseDe_M(A_{aux}\ C_{base})$	$insertarSubclaseDe(C_{base}\ C_{aux})$
3	$subaclaseDe_M(A_{aux}\ C_{base})$	$insertarSubclaseDe(C_{base}\ C_{aux})$

Fuente: (Suero, 2012)
Elaboración: (Suero, 2012)

2.5.3 FAGI-gis: fusing geospatial RDF data (Fusión de datos GEOESPACIAL)

FAGI-gis es una herramienta que se centra en propiedades geoespaciales de entidades integradas, permite realizar la alineación de ontologías, tomando como entrada la obtención de datos a través de consultas SPARQL, y un grupo de enlaces que integran las entidades entre los conceptos, presentando como resultado una nueva incorporación de datos, en donde cada entidad se alinea en una sola. Esta herramienta permite trabajar con la integración de datos geoespaciales geométricos, para la exploración de la información geográfica y explotación de información geoespacial; analiza y recomienda medidas adecuadas sobre la alineación, lanzando un resultado de cómo es almacenada la información geométrica en unión de los metadatos ("Geospatial Semantic Web Community Group", 2015).

CAPITULO III TÉCNICAS DE MINERÍA ONTOLÓGICA

Existen diferentes técnicas de minería ontológica; que permiten extraer información a un conjunto de ontologías existentes, con el fin de ampliar el dominio del conocimiento, en el presente capitulo se detalla de manera precisa cómo funcionan estas técnicas a través de ejemplos sobre algoritmos de alineación, enlazado y mezclado de ontologías.

3.1 Alineación de ontologías

La técnica de alineación de ontologías se da a través del proceso de correspondencias semánticas a través de dos ontologías, permite construir y definir la similitud entre todos los nodos pertenecientes que cumplen el proceso de análisis, obteniendo como resultado las equivalencias entre entidades de las diferentes estructuras semánticas (Zagal, 2008).

Una alineación ontológica es un grupo de correspondencias semánticas que encuentra las relaciones entre entidades que pertenecen a dos o más ontologías, siendo la relación que sostiene diferentes clases, individuos, propiedades o formulas (Aguilar, 2015).

Según menciona (Aguilar, 2015) la alineación de ontologías se conforma por:

- **Un grupo de ontologías O1 y O2:** En cada ontología se procesan los nodos (conceptos), propiedades de los nodos y la jerarquía de los mismos.
- **Un conjunto de p parámetros:** Incorpora los requerimientos para poder elaborar la alineación, como por ejemplo el vocabulario de idioma, cantidad de elementos.
- **Conjunto de recursos R para alineación:** Trabaja con todos los datos de computación y matemática, que son utilizados para elaborar las equivalencias, donde r= tabla de valores de similitud de los algoritmos.
- **Función de alineación f, retorna un conjunto de correspondencias A':** La función f extrae las equivalencias entre los nodos o conceptos.
  El conjunto A': representa las similitudes semánticas descubiertas.

3.1.1 Algoritmo de alineación utilizando KNN-Vecinos y AdaBoost

Se ha elegido trabajar con **KNN-Vecinos,** porque se ha comprobado que este algoritmo se ha desempeñado en varios estudios de agrupación de datos, utiliza una técnica de clasificación, que permite valorar funciones de patrones con respecto a conceptos o información de entrada; trabaja en base a patrones definiendo el vecino más cercano, clasificando los datos en puntos de un plano cartesiano, con el fin de agrupar los valores de las similitudes obtenidas en las matrices de los conceptos pertenecientes a las clases, propiedades y superclases, la clasificación de estos valores de similitud se dan en dos tipos que son: similitud fuerte y similitud débil. KNN-Vecinos utiliza un umbral de pertenencia el mismo que se lo define en base a la experiencia que se tenga sobre datos semánticos,

además este algoritmo aplica la distancia euclidiana (Martin, 2015); descrita en la siguiente ecuación:

$$de = (P1, P2) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Ecuación 1: Distancia euclidiana
Fuente: (Martin, 2015)

En donde P1 y P2 son los dos puntos a analizar.

y2 y x1 similitudes de la ontología 1

y2 y y3 similitudes de la ontología 2

Además se selecciona el algoritmo **AdaBoost** por su gran desempeño en cuanto a clasificación de datos débiles, trabaja con un método que se encarga de la construcción de clasificadores, es decir combina un conjunto de registros simples dando un peso mayor a los datos mal clasificados, y un peso menor a los elementos que se encuentran bien clasificados, AdaBoost utiliza la técnica de precisión, en donde su fuerte es clasificar todos los valores débiles, maneja algunas principales ideas como: seleccionar un conjunto de hipótesis y creando una única hipótesis fuerte final, otro valor significativo es que permite medir el nivel de importancia de cada hipótesis y combinar las mismas de manera perfecta(Uris & Galar, 2015). Este algoritmo utiliza una función de característica para cada dato de entrada y umbral que permite determinar o diferenciar los valores positivos de los negativos (Cantador, 2005), visualizando la figura 9:

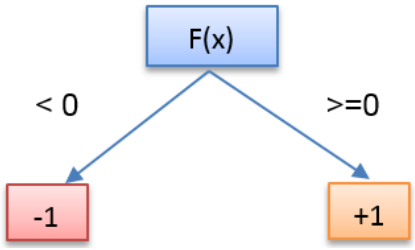

Figura 9: Función de clasificación adaboost
Fuente: (Cantador, 2005)
Elaboración: (Cantador, 2005)

Se realiza la unión de los dos algoritmos, obteniendo un solo método de alineación denominado algoritmo de alineación utilizando KNN-Vecinos y AdaBoost. Primero trabaja en base a los resultados finales obtenidos en las tres matrices de conceptos pertenecientes a las dos estructuras semánticas, la primera matriz es la tabla de afinidad a través de la similitud léxica para clases, una vez que se obtiene este parámetro se procede a la segunda y tercera matriz de afinidad de la similitud semántica entre las propiedades y súper clases de

las ontologías, para poder ver detalladamente el desarrollo de las matrices antes mencionadas dirigirse al capítulo 4.

Una vez obtenidas las matrices de conceptos el algoritmo procesa todos estos datos, y los empieza a clasificar en correspondencias fuertes y débiles, dando una importancia a las correspondencias fuertes ya que estos valores son los que determinan la tabla final de alineación de las dos ontologías.

La Figura 10 muestra el diagrama de flujo de KNN-vecinos y AdaBoost, para la alineación de dos ontologías.

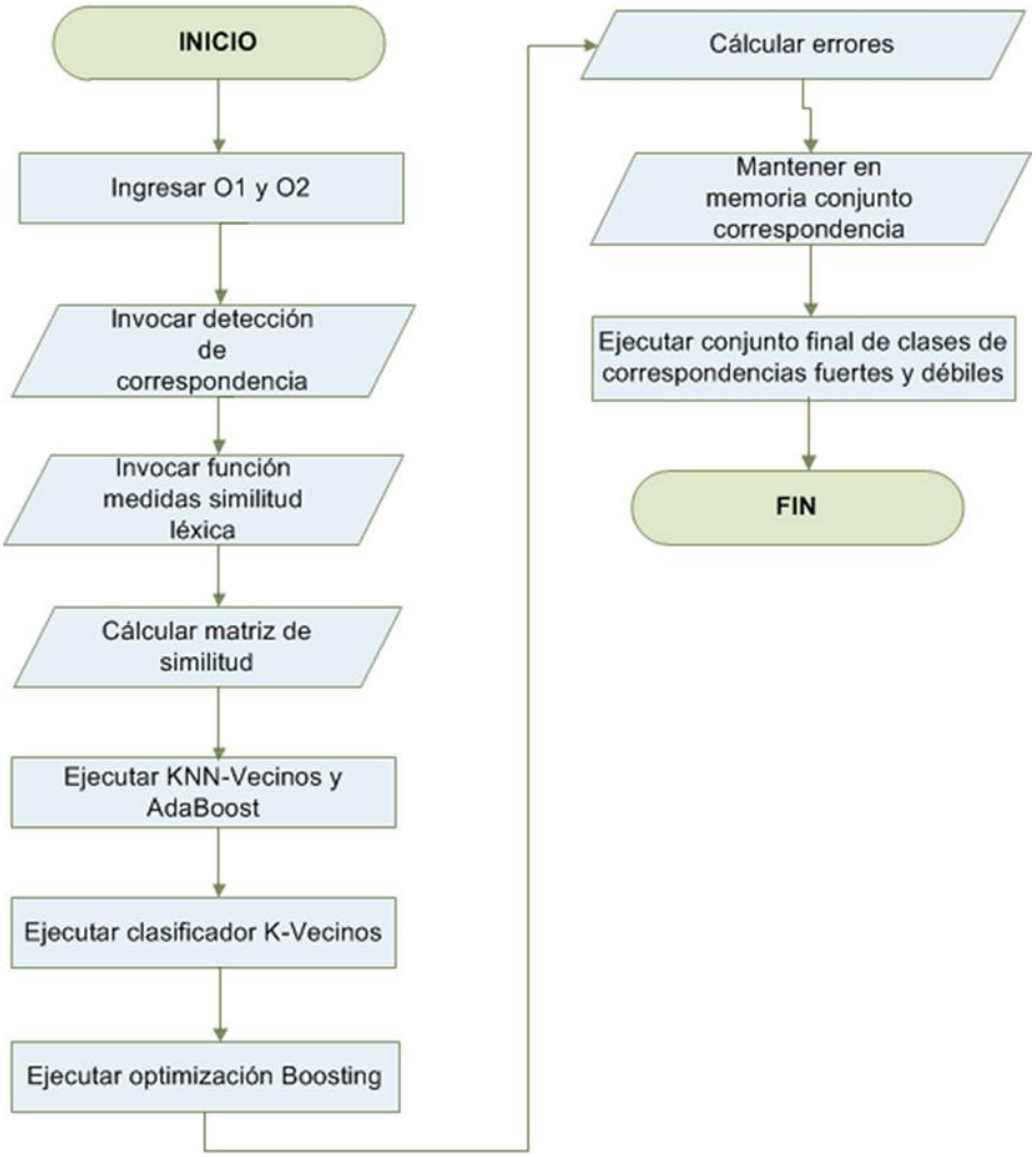

Figura 10: Algoritmo de Alineación utilizando el método de Boosting
Fuente: (Zagal, 2008)
Elaboración: (Zagal, 2008)

3.2 Enlazado de ontologías

La técnica de enlazado de ontologías desarrolla un proceso de conectividad entre estructuras semánticas, sin requerir que las mismas sean mezcladas, permite encontrar relaciones entre entidades que son o pertenecen a diferentes ontologías, se elabora un conjunto de reglas, permitiendo realizar consultas que puedan obtener toda la información de los datos semánticos. Cuando se desarrolla el enlazado se genera como resultado un grupo o conjunto de reglas sobre los datos semánticos, determinando las correspondencias; el enlazado de las estructuras semánticas se puede considerar como un proceso matemático de intersección de conjuntos. Por ejemplo se obtiene dos ontologías y luego se genera una tercera ontología que permitirá la navegación entre las mismas que se están enlazando (Rangel, 2015). A continuación se despliegan los tipos de enlazados de ontologías

3.2.1 Enlazado Débil de Ontologías.

El enlazado débil no forma nuevos nodos o conceptos, utiliza la técnica de analizar las ontologías, una vez que las analiza presenta hipótesis especificas en cada una, es decir permite guiarnos a un nuevo resultado (Rangel, 2015). Con el enlazado débil podemos crear enlaces directos, adquiriendo equivalencias entre conceptos, permitiendo generar un grupo de reglas entre las mismas.

Ejemplo de enlazado débil de ontologías:

La Figura 11, indica dos ontologías que aplican la técnica de enlazado débil, con los nombres de OA (ontología A) y OB (ontología B) obteniendo como un resultado débil una tercera ontología con el nombre OC (ontología C); en donde se observa la equivalencia entre dos nodos que son el concepto 8 (C8) y el concepto C (Cd) lo cual es una relación o igualdad entre las estructuras semánticas “isEquivalent”.

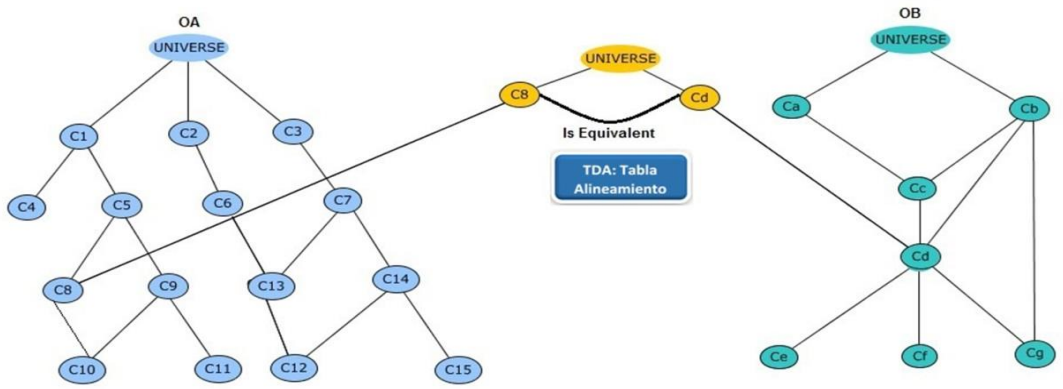

Figura 11: Enlazado débil de OA y OB
Fuente: (Rangel, 2015)
Elaboración: (Rangel, 2015)

3.2.2 Enlazado Fuerte de Ontologías.

La técnica del enlazado fuerte se elabora de forma manual o semi-automática, se apoya del experto que se está enlazado, generando la ontología de intersección, lo cual significa que se definen nuevos nodos o conceptos, en donde se obtiene como resultado final la creación de una tercera ontología con partes de conocimiento enlazado (Rangel, 2015).

La Figura 12 enlaza de manera fuerte dos ontologías, con el nombre de OA (ontología A) y OB (ontología B), obteniendo la creación de la intersección entre las mismas con ayuda del experto del conocimiento global, en donde se puede observar los nuevos nodos creados por un experto “&”,”$”.

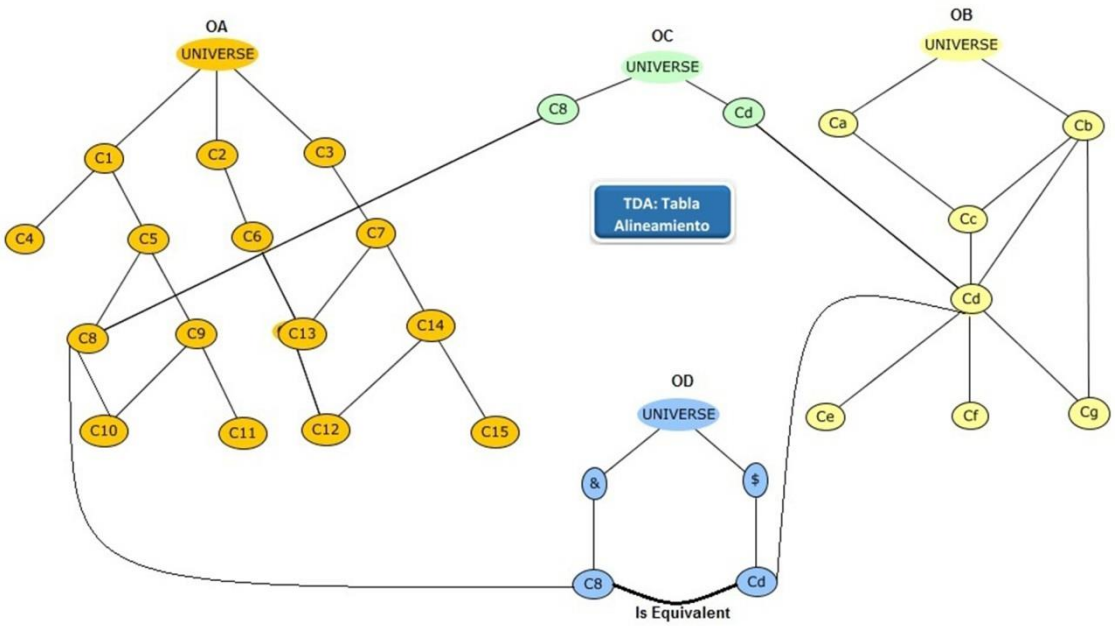

Figura 12: Enlazado Fuerte de OA y OB
Fuente: (Rangel Piña, 2015)
Elaboración: (Rangel Piña, 2015)

3.2.3 Diagrama de flujo del algoritmo Enlazado General.

La Figura 13, describe la funcionalidad del algoritmo general para poder realizar el enlazado de ontologías, se inicia el algoritmo alineando las dos estructuras semánticas, y el resultado se lo va almacenando en una tabla o en una lista de arreglos, se define el porcentaje de similitud de estos valores; si los datos de la matriz que se obtiene no es vacía se procede a crear una nueva ontología vacía, luego se procede a crear todos los conceptos que se encuentren en el arreglo y finalmente se agregan las equivalencias de los conceptos alineados en la tabla final de alineación.

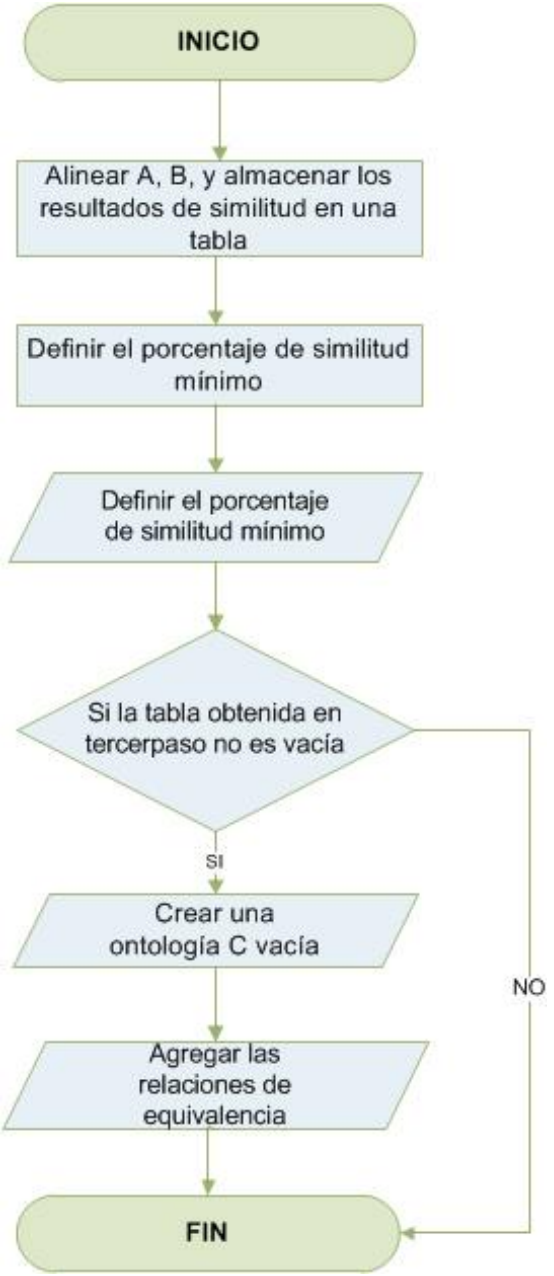

Figura 13: Algoritmo de enlazado general
Fuente: (Rangel, 2015)
Elaboración: (Rangel, 2015)

3.3 Mezcla de ontologías

El mezclado de ontologías es otra de las técnicas de la minería ontológica, su proceso consiste en unir varias ontologías entre el mismo dominio para establecer el conocimiento, con diferentes representaciones y en donde las estructuras semánticas pueden o no concordar en ciertos nodos o conceptos. La mezcla permiten tener todo el conocimiento de manera local, esto es muy útil en los sistemas distribuidos ya que permite consultar en el mismo dominio facilitando la información y evitando consultas en varios sistemas, al momento de aplicar esta técnica se necesita la presencia de un experto del conocimiento el

mismo que debe estar presente al momento de la ejecución para tomar las decisiones. (Aguilar, 2015).

A continuación se detallan los diferentes tipos de Mezclas de ontologías que existen

3.3.1 Mezcla débil de ontologías

Para desarrollar la técnica del mezclado débil, se debe tener como entrada dos ontologías, en donde se tiene que encontrar cada nodo o concepto de una ontología A en una ontología B, en caso de que la respuesta sea positiva y se encuentre el nodo en B, se determina al proceso como completo y se inserta todas las propiedades y equivalencias que pertenezcan al nodo o concepto de la ontología B; hay que tener precaución al momento de realizar esta mezcla ya que se puede generar duplicaciones, datos disjuntos e incompatibilidades. Como resultado obtenemos una ontología C que puede ser observado como la unión parcial de dos conjuntos (Rangel, 2015).

La Figura 14, indica un ejemplo de mezclado débil de estructuras semánticas, donde no es dejado el conocimiento por fuera; se observa que el mismo es extraído de la ontología B (OB) para ser agregado en la ontología A (OA), donde solo son obtenidos los conceptos que se encuentran alineados en la ontología B. Si logra existir un correcto alineamiento serán obtenidos todos los conceptos de OB.

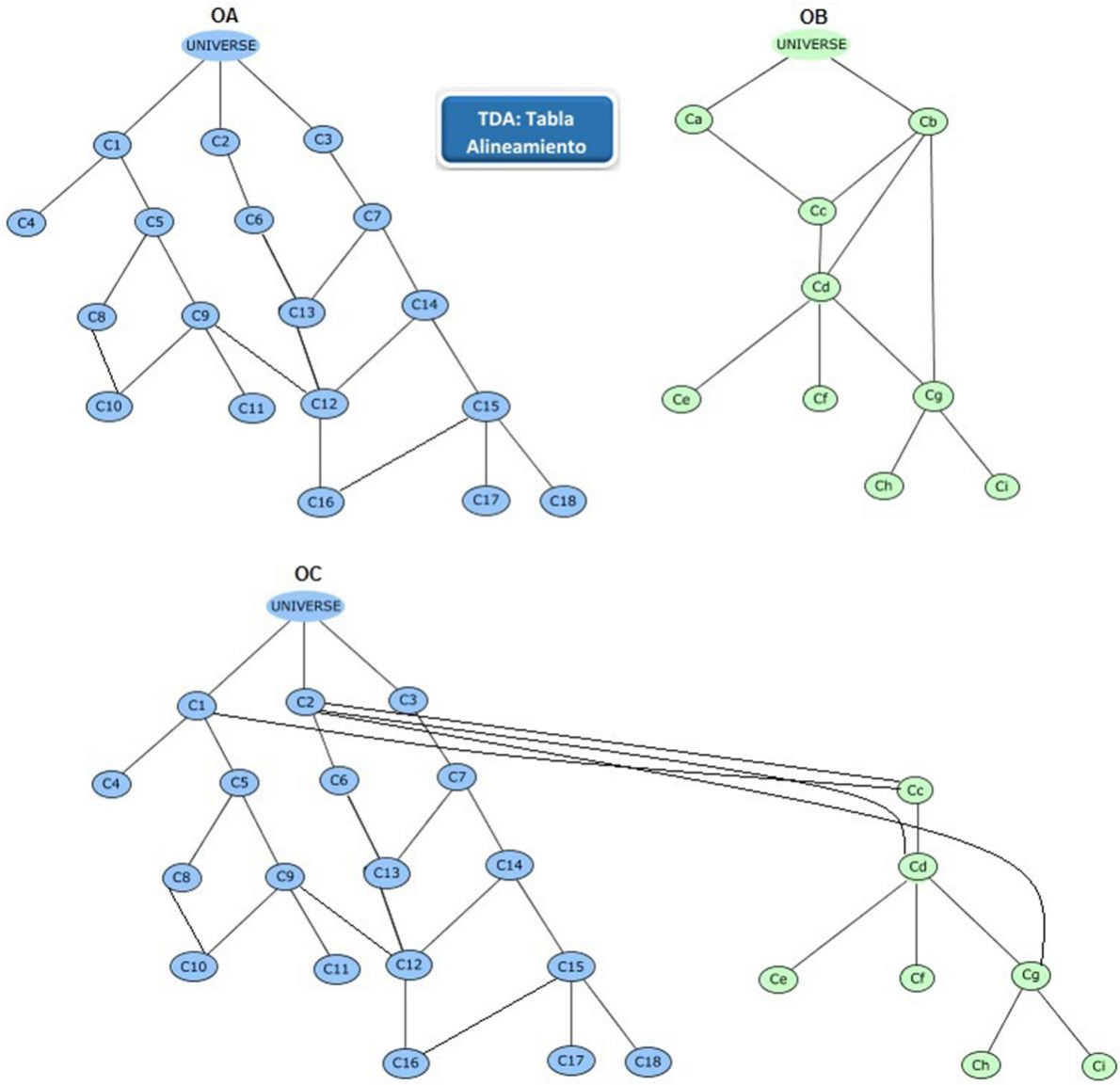

Figura 14: Mezcla débil donde no es dejado conocimiento por fuera
Fuente: (Rangel, 2015)
Elaboración: (Rangel, 2015)

La Figura 15, muestra otro ejemplo de mezcla débil donde un concepto de ontología B (OB) no es agregado en un concepto de ontología C (OC), ya que no todos los nodos de OB fueron alineados, por lo que esto hace que un nodo o concepto no logre ser incluido.

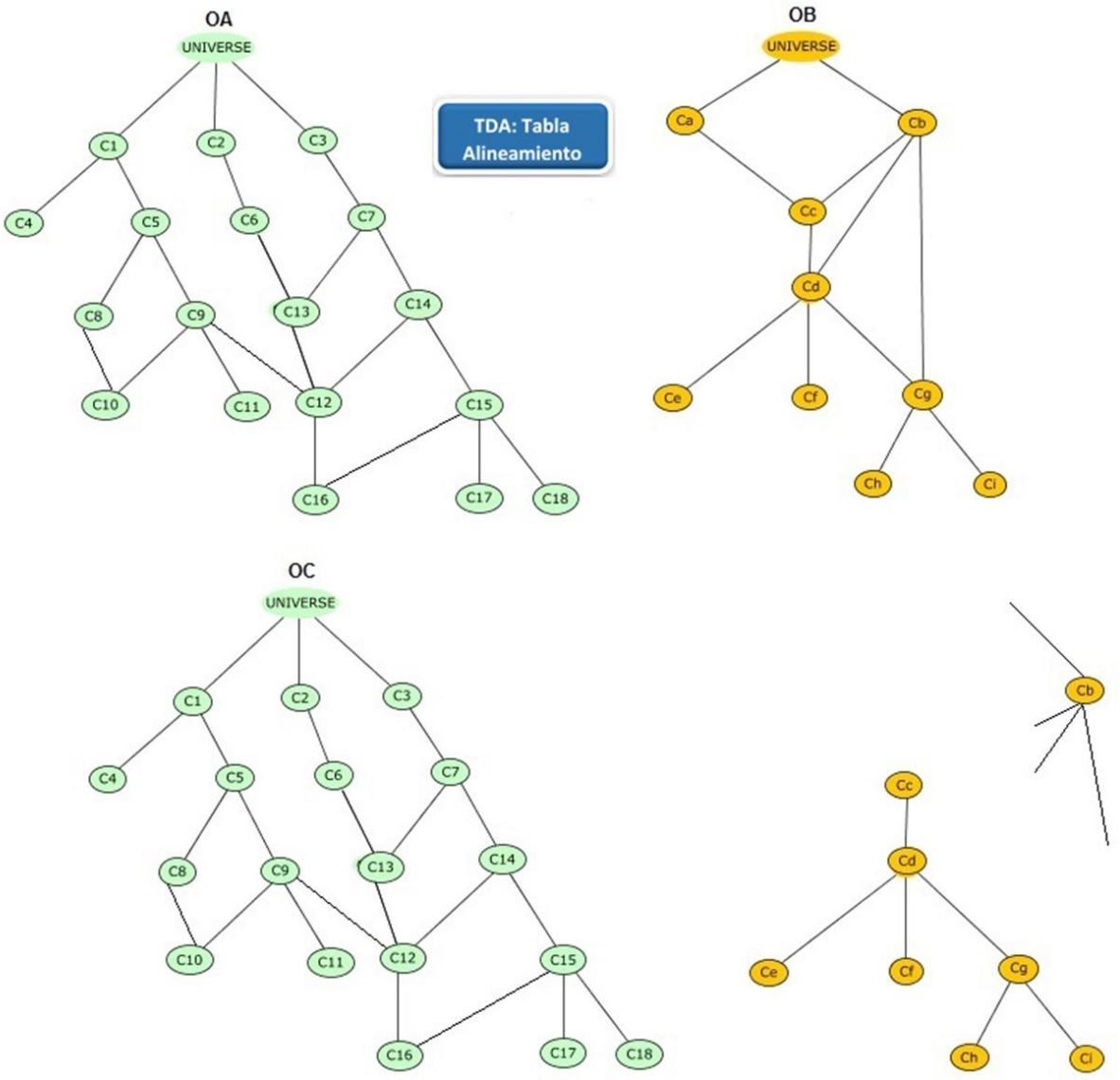

Figura 15: Mezcla débil donde un concepto B no es agregado en un concepto C
Fuente: (Rangel, 2015)
Elaboración: (Rangel, 2015)

3.3.2 Diagrama de flujo del algoritmo Mezcla Débil

La Figura 16, muestra el proceso del algoritmo de mezcla débil, inicia copiando la ontología A (OA) en la ontología C (OC) por lo que se requiere el tener una tabla de alineamiento en donde las dos estructuras semánticas sean consistentes presentando una correcta alineación, una vez que se tiene esta tabla se recorre en orden los hijos que presenta la misma, si en caso no existe algún hijo (alineación) este es añadido y se agregan las relaciones con el padre. Caso contrario si el hijo y padre si fueron alineados con su equivalente y la relación es diferente se procede a copiar la relación, se termina el proceso y finaliza el algoritmo.

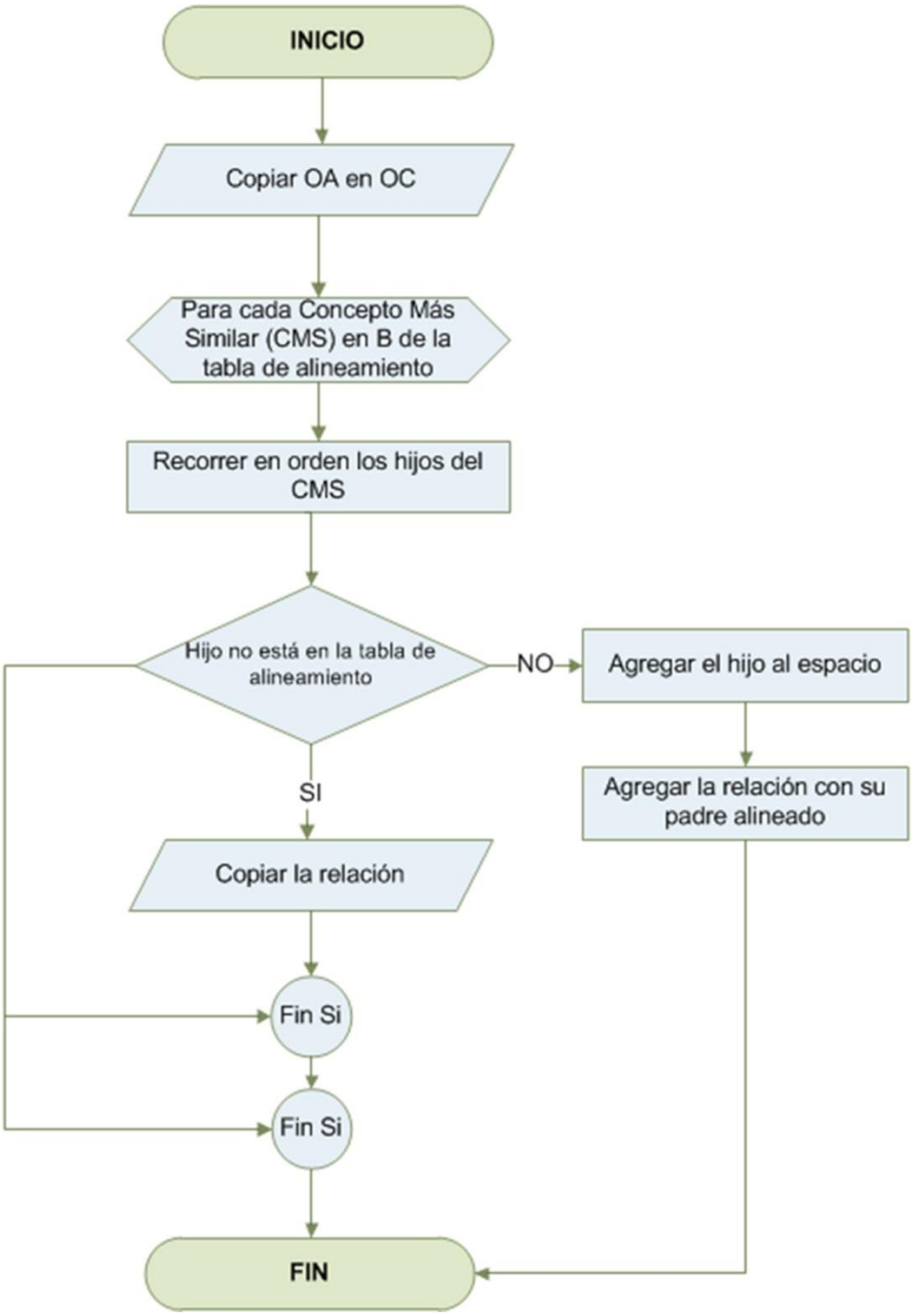

Figura 16: Algoritmo de mezcla débil
Fuente: (Rangel, 2015)
Elaboración: (Rangel, 2015)

3.3.3 Mezcla fuerte de ontologías

La mezcla fuerte es una técnica de ontologías que se compone de una mezcla débil, pero con la incorporación del conocimiento dejado por fuera de la ontología B, en caso de que no estén incorporados los nodos la mezcla fuerte se encarga de incorporarlos en la nueva ontología C, y de agregar las relaciones de todos los conceptos que no fueron incorporados en la mezcla débil, con los nodos que fueron copiados o alineados (Rangel, 2015).

La Figura 17 indica un ejemplo de mezclado fuerte, en donde el concepto de ontología B (CB) es agregado en el concepto de ontología C, todos los conceptos de CB fueron alineados, mientras que los nodos que no cumplieron con el proceso de alineación no se copiaran en la ontología resultado, para poder realizar esta técnica se debe incorporar todos los conceptos alineados en la ontología C copiando las relaciones que no han sido alineadas o insertadas.

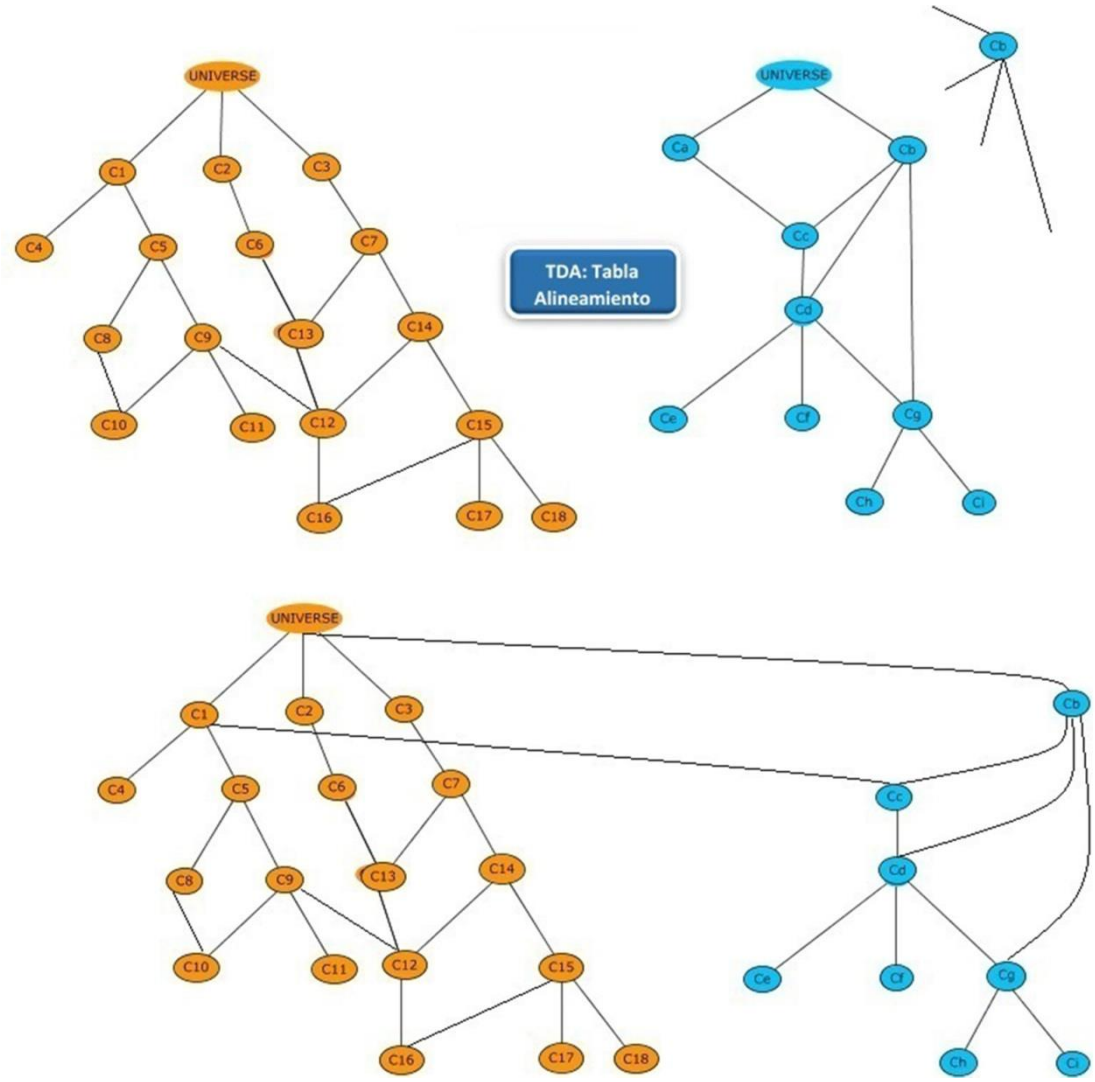

Figura 17: Mezcla fuerte donde se incorpora un concepto de B que no fue agregado a C
Fuente: (Rangel, 2015)
Elaboración: (Rangel, 2015)

3.3.4 Diagrama de flujo del algoritmo Mezcla Fuerte

La Figura 18 muestra el algoritmo de mezcla fuerte, es una técnica bastante robusta está compuesta de la mezcla débil, detectando los conceptos de la segunda ontología (OB) que no han sido copiados o alineados, en caso de que se detecte que no se han alineado al menos un subárbol busca el antecedente más cercano de los conceptos copiados, copia todo el subárbol en el espacio, creando y copiando las relaciones con el antecedente entre

los conceptos del subárbol que no fueron copiados; para cada nodo del subárbol que no fue copiado se vuelve a recorrer en orden como se lo hace en la mezcla débil, caso contrario si para cada hijo del concepto OB que está en la tabla de alineamiento, se crea la relación del padre del subárbol no copiado al hijo alineado en la ontología OC resultado, si el concepto a ser copiado no va en el padre universo, automáticamente se pasa como hijo del ancestro (padre) actual, si el concepto ya se encuentra en la tabla de alineamiento, no se debe realizar ningún procedimiento más ya que las relaciones están copiadas, dando por finalizado el algoritmo.

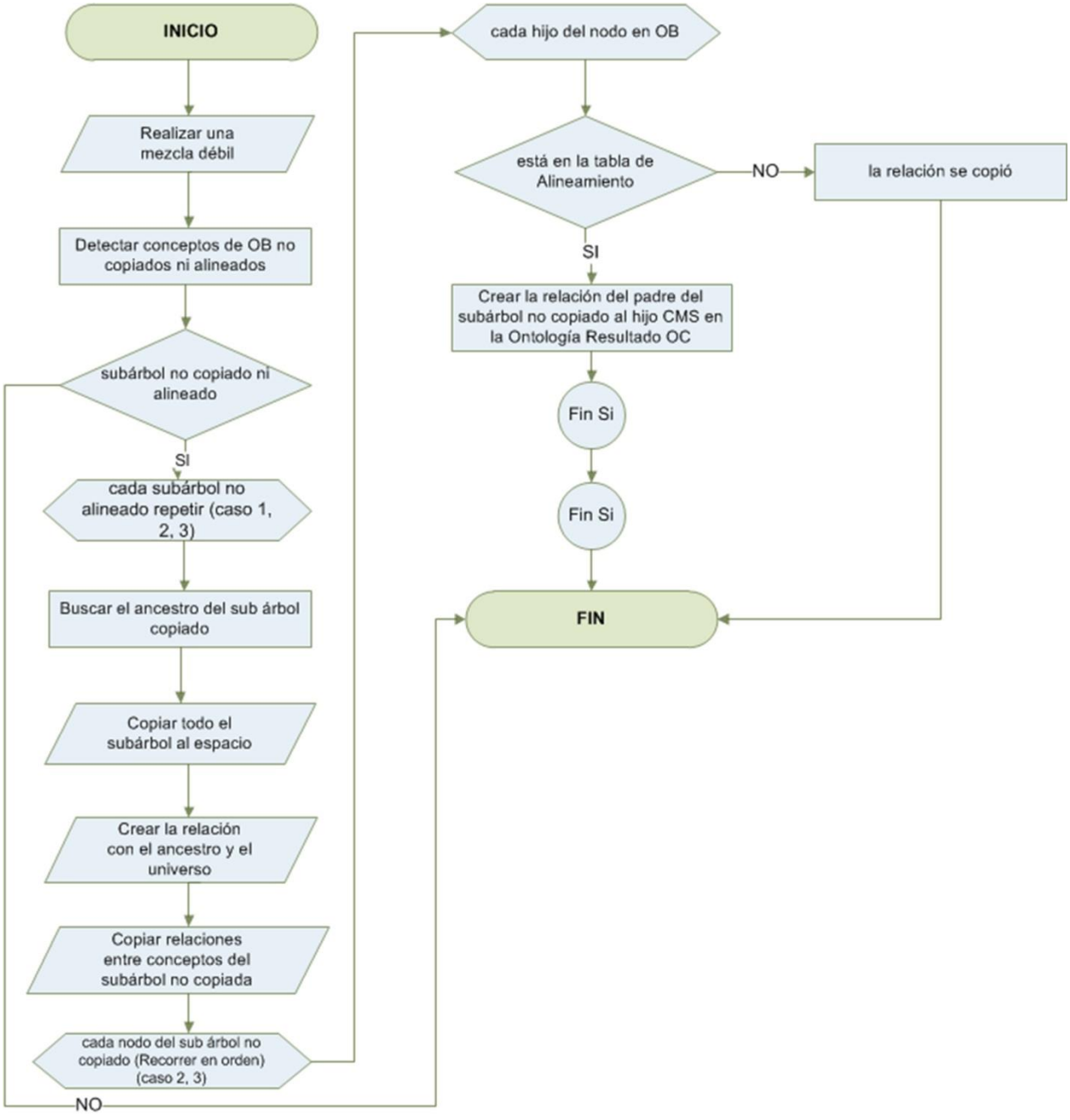

Figura 18: Algoritmo de mezcla Fuerte
Fuente: (Rangel, 2015)
Elaboración: (Rangel, 2015)

CAPÍTULO IV: IMPLEMENTACIÓN DE TECNICAS DE MINERIA DE DATOS

En este capítulo se implementa las diferentes técnicas de minería ontológica, se menciona que ontologías (archivos owl) se van a trabajar, se realiza la implementación de las tablas de similitudes para las ontologías, además se aplica la alineación, enlazado y mezclado de ontologías.

4.1 Búsqueda y Selección de recursos ontológicos

Para el desarrollo de este trabajo se ha buscado dos ontologías que tienen el nombre de Culture-shop y Library en un tipo de formato owl, presentan una completa estructura semántica, las mismas que pertenecen a un dominio educativo y se encuentran disponibles para su uso en el libro Ontology Matching second edition (Alineación de Ontologías segunda edición) por el autor (Euzenat & Shvaiko, 2013), quien trabaja en proyectos de Web Semántica específicamente en el tema de alineación de ontologías. Estas estructuras semánticas que han sido seleccionadas pertenecen al mismo dominio, pero con diferente nomenclatura semántica; no presentan un vocabulario muy extenso y son de fácil comprensión debido a su tamaño, han sido revisadas y analizadas en la herramienta Protegé, además estas ontologías han sido probadas y ejecutadas sobre una aplicación web que ha sido desarrollada para aplicar todas las técnicas de la minería ontológica, obteniendo como resultado una tercera ontología bastante extensa y robusta.

4.1.1 Culture-shop

La ontología Culture-shop pertenece a un dominio educativo, está conformada por un vocabulario de términos reglas y relaciones que admiten extender su vocabulario, permite modelar un anexo técnico sobre un proyecto en donde incluye cierta información acerca de productos educativos, personas etc. Presenta varias clases principales, para un mayor entendimiento se elabora la tabla 3 sobre las clases que conforman la estructura semántica.

Tabla 3: Clases de la ontología Culture-shop

Culture-shop	
Clase Original	**Traducción**
Book	Libro
CD	CD
Children	niños
DVD	DVD
Person	Persona
Pocket	Bolsillo
Popular	Popular
Product	Producto
Publisher	Editor
Science	Ciencia
Textbook	Libro de Texto

Fuente: (Euzenat & Shvaiko, 2013)

Elaboración: (Euzenat & Shvaiko, 2013)

La Figura 19, muestra la ontología leída desde la herramienta Protegé.

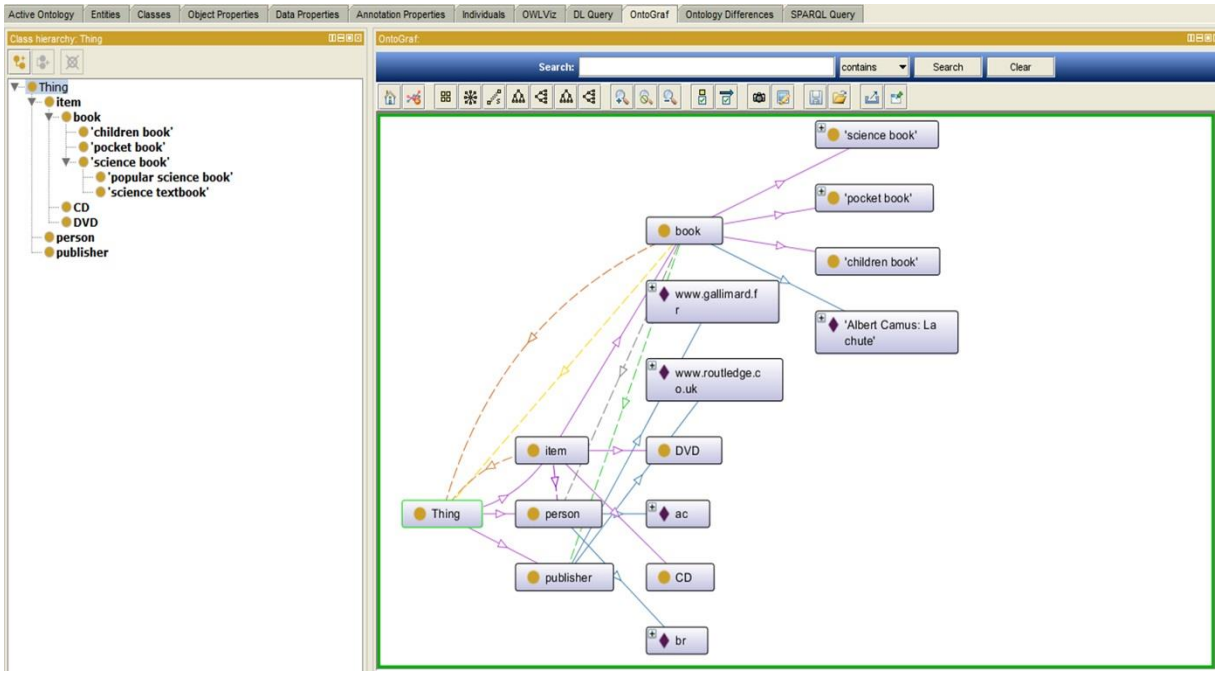

Figura 19: Ontología Culture-shop
Fuente: (Euzenat & Shvaiko, 2013)
Elaboración: (Euzenat & Shvaiko, 2013)

4.1.2 Library

Library pertenece al dominio de educación al igual que la ontología Culture-shop, estas dos ontologías son bastante similares en sus clases pero con propiedades y superclases totalmente diferentes, se conforma por 11 clases principales, con relaciones, reglas y términos en su vocabulario, formando conocimiento sobre la educación a través de bibliotecas. La tabla 4 indica todas las clases de esta estructura semántica.

Tabla 4: Clases de la ontología Library

Library	
Clase Original	**Traducción**
Autobiography	Autobiografía
Biography	Biografía
Essay	Ensayo
Human	Humano
LiteraryCritic	Literatura critica
Literature	Literatura
Novel	Novela
Poetry	Poesía
Politics	Políticas
Volume	Volumen
Writer	Escritor

Fuente: (Euzenat & Shvaiko, 2013)
Elaboración: (Euzenat & Shvaiko, 2013)

La Figura 20 muestra la ontología Library revisada desde la herramienta Protegé.

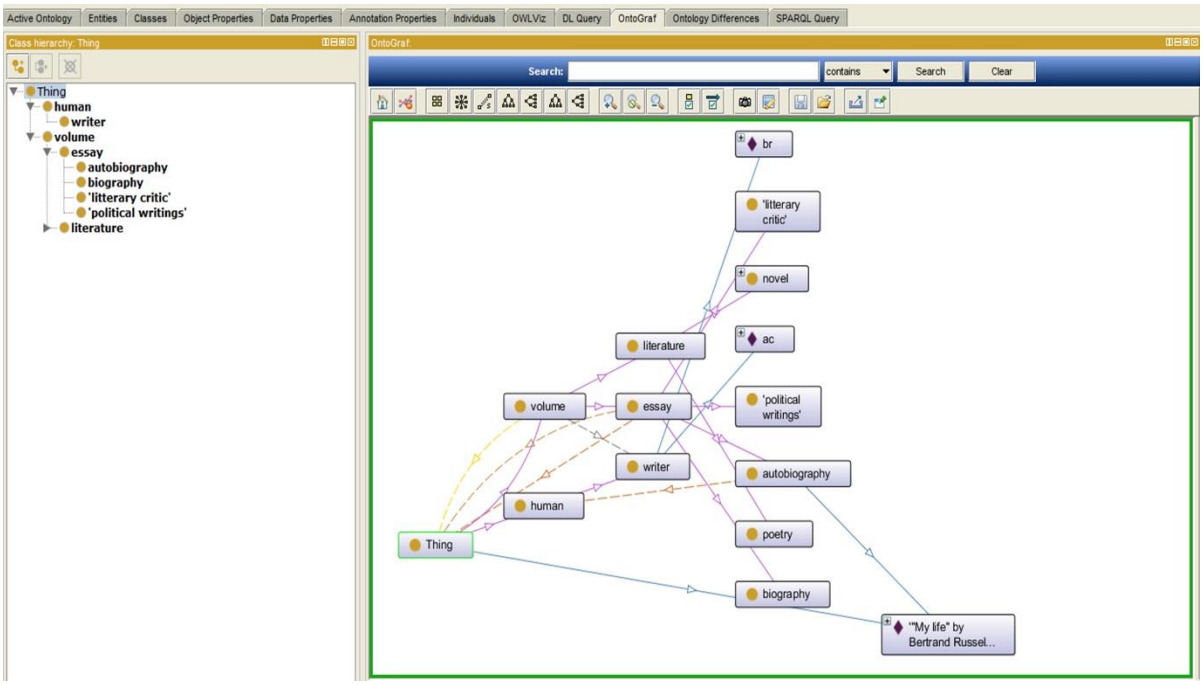

Figura 20: Ontología Library
Fuente: (Euzenat & Shvaiko, 2013)
Elaboración: (Euzenat & Shvaiko, 2013)

4.1.3 Propiedades de la Ontología Culture-shop

La ontología Culture-shop pertenece al ámbito educativo, permite compartir y reutilizar el conocimiento existente en las técnicas de minería ontológica, se obtienen 10 Data Properties (Propiedades de datos) presentadas en la tabla 5.

Tabla 5: Propiedades de la ontología Culture-shop

Culture-shop	
Propiedad Original	**Traducción**
Contributor	Contribuyente
Creator	Creador
Date	Fecha
Description	Descripción
Firtsname	Primer Nombre
Id	Id
Lastname	Apellido
Name	Nombre
Price	Precio
Topic	Tema

Fuente:(Euzenat & Shvaiko, 2013)
Elaboración: (Euzenat & Shvaiko, 2013)

4.1.4 Propiedades de la ontología Library.

Las propiedades de la ontología Library tienen cierta similitud con Culture-shop, se conforma por 8 Data Properties (Propiedades de datos). Esta estructura semántica admite reutilizar el conocimiento existente para las técnicas de minería ontológica, con el propósito de enriquecer las estructuras semánticas, se indica la tabla 6 sobre las propiedades de esta ontología.

Tabla 6: Propiedades de la ontología Library

Lybrary	
Propiedad Original	**Traducción**
Contributor	Contribuyente
Creator	Creador
Date	Fecha
Description	Descripción
Name	Nombre
Title	Titulo
Year	Año
Year	Año

Fuente:(Euzenat & Shvaiko, 2013)
Elaboración: (Euzenat & Shvaiko, 2013)

4.1.5 SuperClases de la ontología Culture-shop

Las superclases son clases padre de las subclases que presentan las ontologías. Culture-shop presenta superclases muy distintas a la ontología Library, las mismas que han sido obtenidas en base a la siguiente consulta SPARQL:

PREFIX a: http://book.ontologymatching.org/example/culture-shop.owl#
SELECT * FROM <http://book.ontologymatching.org/example/culture-shop.owl#> WHERE {a:Book rdfs:subClassOf ?s} (Ossowski, 2014)

Obteniendo como resultado los datos de la tabla 7.

Tabla 7: SuperClases de la ontología Culture-shop

Culture-shop	
SuperClases Originales	**Traducción**
Product	Producto
Book	Libro
Science	Ciencia

Fuente: (Euzenat & Shvaiko, 2013)
Elaboración: (Euzenat & Shvaiko, 2013)

4.1.6 SuperClases Library

Se obtienen 4 superclases de la ontología Library, de acuerdo a la siguiente consulta SPARQL:

PREFIX a: <http://book.ontologymatching.org/example/library.owl#>

SELECT * FROM <http://book.ontologymatching.org/example/library.owl#> WHERE {a:Biography rdfs:subClassOf ?s}

Presentando resultados distintos a la ontología Culture-shop, como lo indica la tabla 8.

Tabla 8: SuperClases de la ontología Library

Library	
SuperClases Originales	**Traducción**
Essay	Ensayo
Volume	Volumen
Literature	Literatura
Human	Humano

Fuente: (Euzenat & Shvaiko, 2013)
Elaboración: (Euzenat & Shvaiko, 2013)

4.2 Carga de Ontologías en Virtuoso

Para poder realizar la carga de las ontologías en el servidor de virtuoso, es requisito que los archivos sean formato owl, se ingresa en la aplicación y se selecciona el menú cargar ontologías, elegir la opción examinar desplegándose una ventana de navegación, que permite seleccionar las ontologías desde cualquier parte del computador, como se indica en la Figura 21; véase Anexo 1 con el completo desarrollo.

Figura 21: Carga de ontologías
Fuente: El autor
Elaboración: El autor

Una vez ya seleccionada la ontología, se debe ingresar un nombre de referencia, que sirve como identificativo, finalmente dar clic en el botón subir que es encargado de guardar la ontología en el servidor; si la carga es correcta aparecerá un mensaje de archivo almacenado, como se indica en la Figura 22.

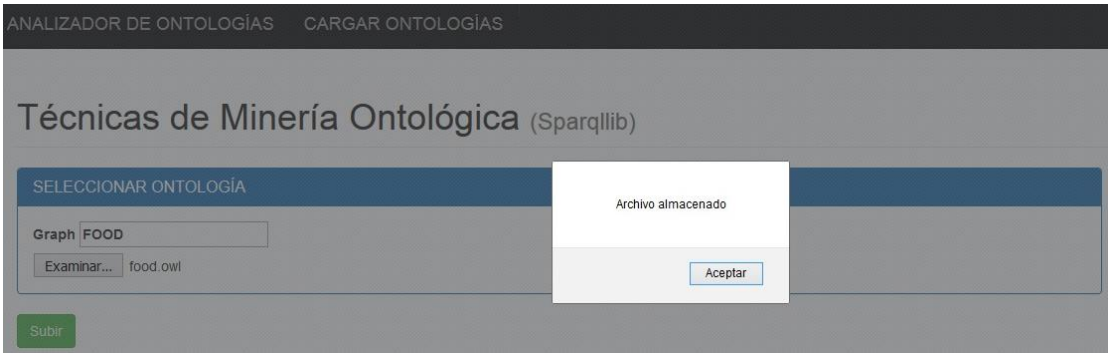

Figura 22: Carga exitosa de ontologías
Fuente: El autor
Elaboración: El autor

En la Figura 23, se indica un mensaje de error en la carga, esto se da cuando el archivo no tiene el formato correcto, por ende no se puede almacenar.

Figura 23: Carga no exitosa de ontologías
Fuente: El autor
Elaboración: El autor

4.3 Implementación de Alineación entre dos ontologías

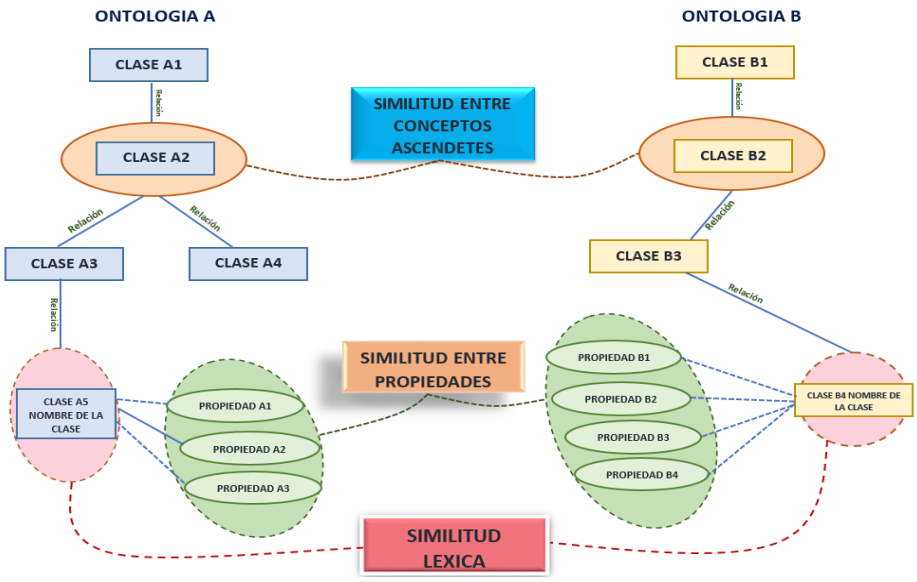

Figura 24: Elementos de ontologías con sus medidas de similitud
Fuente: (Zagal, 2008)
Elaboración: (Zagal, 2008)

La Figura 24, muestra el esquema que se utiliza para la alineación de dos ontologías, determinando el desarrollo de las medidas de similitud, clasificadas en: medidas de similitud con base en términos y medidas de similitud semántica.

Medidas de similitud con base en términos: trabaja en los nombres de los objetos como clases y propiedades que pertenecen a una ontología.

Medidas de similitud semántica: se centra en los objetos de la semántica como la similitud entre propiedades de clases y la similitud entre superclases de las clases.

- Similitud entre propiedades de clases: revisa las equivalencias que existen entre las propiedades de ambas clases.
- Similitud entre superclases: compara y revisa las dos superclases que tengan la mayor similitud (Zagal, 2008).

La similitud léxica, entre propiedades y superclases se utiliza como enfoque preciso para poder desarrollar una alineación correcta. Para mayor conocimiento sobre el funcionamiento de las similitudes se hace referencia en las secciones 4.3.2 4.3.3 y 4.3.4

4.3.1 Tabla o Matriz de similitud

Se elabora una tabla por cada similitud. La tabla o matriz de similitud, se ejecuta mediante la siguiente estructura.

- X = total de clases de OA.
- Y = total de clases de OB.
- Z = (X) (Y), formando el tamaño de la tabla de similitud.
- Simi (C_{oa}, C_{ob}), medida de similitudes.
- (A_{oan}, C_{obn}), clases de las ontologías a ser comparadas.

En la tabla 9 se muestra el resultado final de la combinación de similitudes.

Tabla 9: Similitudes General

Simi (C_{oa}, C_{ob})	C_{ob1}	C_{ob2}	C_{ob3}	C_{ob4}
C_{oa1}				
C_{oa2}				
C_{oa3}				
C_{oa4}				

Fuente: (Zagal, 2008)
Elaboración: (Zagal Flores, 2008)

4.3.2 Similitud Léxica

La similitud léxica se realiza en base a la distancia de Levenshtein, fue desarrollada en el año de 1965 por el científico de origen ruso de nombre Vladimir Levenshtein (García, 2015), estableciendo la cantidad mínima de caracteres o cálculos que se requieren para poder convertir una cadena de caracteres en otra, las operaciones de edición que utiliza este método son:

- Inserción de un carácter.
- Eliminación de un carácter.
- Substitución de un carácter.

La distancia de Levenshtein se calcula mediante la similitud directa que son las operaciones de edición aplicadas de la siguiente manera formal:

- Conjunto de caracteres u operaciones en cadenas Op (Op: S --> S)
- Función de costo w; esta función permite transformar la primera cadena en una segunda cadena y viceversa *&(s, t)*.

En donde &(s, t) es la secuencia con menor costo de las operaciones que permiten transformar s en t (García, 2015).

El cálculo de grado de similitud se calcula con el valor obtenido en la similitud directa en donde se aplica la siguiente ecuación matemática**:**

$$\boldsymbol{Afinidad = 1 - (\frac{D}{L})}.$$

Ecuación 2: Cálculo del grado de afinidad de similitud léxica
Fuente: Cosmico, 2016

- En donde D es el valor de similitud directa (Distancia de Levenshtein).
- L es la longitud de la palabra o clase más larga (Cosmico, 2016).

La distancia de Levenshtein tiene medidas en un rango entre [0,1] en donde si los valores son 1 los dos objetos son iguales, pero si el resultado de los valores es 0 los dos objetos son totalmente diferentes.

A continuación se realiza varios ejemplos, aplicando la distancia de Levenshtein entre dos estructuras semánticas.

4.3.2.1 Ejemplos de Cálculo de similitud léxica

A continuación se presentan algunos ejemplos sobre el cálculo de similitud léxica entre las estructuras semánticas Culture-shop y Library, los mismos que han sido elegidos de manera aleatoria, con la finalidad de explicar y comparar resultados.

Ontologías Culture-shop y Library

Calcular la distancia de Levenshtein, aplicando el valor directo y el cálculo de grado de similitud entre las siguientes clases.

- Book y Biography
- Person y Human
- Book y Novel

Valor Directo Book y Biography:

Book (?) Biography

1. Book (?) Boography (Sustituir ‘i’ por ‘o’)
2. Book (?) Bookraphy (Sustituir de ‘k’ por ‘g’)
3. Book (?) Bookaphy (eliminar ‘r’ después de ‘k’)
4. Book (?) Bookphy (eliminar ‘a’ después de ‘k’)
5. Book (?) Bookhy (eliminar ‘p’ después de ‘k’)
6. Book (?) Booky (eliminar ‘h’ después de ‘k’)
7. Book (?) Book (eliminar ‘y’ después de ‘k’)

   Book (?) Book

El cálculo del valor directo entre Book y Biography es de **7**, por que se utilizan siete operaciones de edición para igualar las dos clases o palabras.

Cálculo de grado de similitud Book y Biography:

$$\boldsymbol{Afinidad = 1 - (\frac{D}{L})}$$

$$Afinidad = 1 - (\frac{7}{9})$$

$$Afinidad = 1 - 0.7777777$$

$$Afinidad = 0.222$$

El grado de similitud entre Book y Biography arroja un resultado de **0.222**, por lo que las dos clases o palabras son un poco diferentes en base a la definición de la similitud léxica, que

explica si el resultado es 1 son totalmente iguales y si el resultado es 0 entonces las palabras son totalmente diferentes.

Valor directo Person y Human:

Person (?) Human

1. Person (?) Puman (Sustituir ‘H’ por ‘P’)
2. Person (?) Peman (Sustituir ‘u’ por ‘e’)
3. Person (?) Peran (Sustituir ‘m’ por ‘r’)
4. Person (?) Persn (Sustituir ‘a’ por ‘s’)
5. Person (?) Person (Insertar ‘o’ antes de ‘n’)

Person (?) Person

El valor directo para las clases Person y Human es de 5, porque se utilizan cinco operaciones de edición para igualar las dos clases o palabras.

Cálculo de grado de similitud Person y Human:

$$\boldsymbol{Afinidad = 1 - (\frac{D}{L})}$$

$$Afinidad = 1 - (\frac{5}{6})$$

$$Afinidad = 1 - 0.8333333$$

$$Afinidad = 0.166$$

Calculando el grado de similitud entre las dos clases Person y Human el resultado es de **0.166**, determinado que estas clases tienen cierta afinidad.

Valor directo Book y Novel:

Book (?) Novel

1. Book (?) Bonel (Sustituir ‘N’ por ‘B’)
2. Book (?) Booel (Sustituir ‘o’ por ‘n’)
3. Book (?) Bookl (Sustituir ‘e’ por ‘k’)
4. Book (?) Book (Eliminar ‘l’ después de ‘k’)

Book (?) Book

El valor directo de las clases Book y Novel es de **4** pasos de edición.

Cálculo de grado de similitud Book y Novel:

$$\boldsymbol{Afinidad = 1 - (\frac{D}{L})}$$

$$\boldsymbol{Afinidad = 1 - (\frac{4}{5})}$$

$$\boldsymbol{Afinidad = 1 - 0.8}$$

$$\boldsymbol{Afinidad = 0.2}$$

El grado de similitud entre Book y Novel es de **0.2**, logrando así concluir que estas clases tienen una afinidad baja.

La tabla 10, indica la similitud léxica de acuerdo a los cálculos realizados en los ejemplos de las ontologías Culture-shop y Library.

Tabla 10: Similitud léxica de ontologías Culture-shop y Library

Simi_lexica (C_{oa}, C_{ob})	Biography	Human	Novel
Book	0.2222	0	0.2
Person	0.1111	0.16666	0

Fuente: El autor
Elaboración: El autor

Las ontologías Culture-shop y Libray de la similitud entre clases con valor directo, se encuentra en el Anexo 2 de forma completa; y así mismo en el Anexo 3 se aprecia el grado de similitud de las clases.

La Figura 25, muestra la función calcular distancia, que obtiene el procedimiento del cálculo de Levenshtein.

```
1. function calcularDistancia($claseOnt1,$claseOnt2){
2.     return
   computeLevenshteinDistance(str_split($claseOnt1),str_split($claseO
   nt2));
3. }
```

Figura 25: Llamado de función distancia Levenshtein
Fuente: El autor
Elaboración: El autor

El cálculo de la distancia de Levenshtein se indica en la Figura 26, se realiza la inserción, sustitución o eliminación de las ediciones entre las clases de dos ontologías.

```
1. function computeLevenshteinDistance($clase1, $clase2) {
2.    for( $i=0;$i<=sizeof($clase1);$i++){
3.          $distancia_clase[$i][0]=$i;
4.    }
5.    for( $j=0;$j<=sizeof($clase2);$j++){
6.          $distancia_clase[0][$j]=$j;
7.    }
8.          for( $i=1;$i<=sizeof($clase1);$i++){
9.             for( $j=1;$j<=sizeof($clase2);$j++){
10.                    if($clase1[$i-1]==$clase2[$j-
   1]){$aux=0;}else{$aux=1;}
11.                     $distancia_clase[$i][$j]=
   minimum($distancia_clase[$i-1][$j]+1,$distancia_clase[$i][$j-
   1]+1,$distancia_clase[$i-1][$j-1]+($aux));
12.               }
13.           }
14.           return $val =
   $distancia_clase[sizeof($clase1)][sizeof($clase2)];
15.   }
```

Figura 26: Función calcular distancia Levenshtein
Fuente: El autor
Elaboración: El autor

La Figura 27, muestra el cálculo del grado de afinidad entre dos clases pertenecientes a ontologías.

```
1. function calcularAfinidad($claseOnt1, $claseOnt2, $distancia){
2.         $longitud=0;
3.         if(strlen($claseOnt1)>strlen($claseOnt2)){
4.             $longitud=strlen($claseOnt1);
5.         }else{
6.             $longitud=strlen($claseOnt2);
7.         }
8.         $afinidad = 1-($distancia/$longitud);
9.         return $afinidad;
10.          }
```

Figura 27: Función calcular grado de afinidad
Fuente: El autor
Elaboración: El autor

4.3.3 Similitud entre propiedades

La similitud entre las propiedades, aplica un cálculo entre los nombres de las clases que existen, se requiere como ingreso la medida de similitud léxica, la función de esta similitud es de verificar la distancia que existe entre el conjunto de propiedades de dos clases, en donde la función principal de esta similitud es lograr reconocer si cada propiedad P1 del conjunto de propiedades P de una clase C1 concuerda con otra propiedad del conjunto P2 del conjunto de propiedades P de otra clase C2; utiliza una correspondencia de similitud

como por ejemplo, suponer que la similitud_propiedad(“Date”, “Date”) sea igual que la similitud_propiedad(“Date”, “Date”) (Likavec, Osborne, & Cena, 2015).

$$S(a,b) = \frac{|A \cap B|}{|AUB|}$$

Ecuación 3: Cálculo de similitud entre propiedades
Fuente: García, 2015.

En donde:

$|A \cap B|$ = intersección de conjuntos

$|AUB|$ = *unión de conjuntos* (García, n.d.)

Para el cálculo de la similitud entre propiedades, es necesario realizar una matriz interna aplicando la distancia de edición entre las dos ontologías, una vez que se obtiene este primer paso, se procede a aplicar la ecuación de la intersección sobre la unión de los conjuntos, en donde la ecuación:

$$(a,b) = \frac{|A \cap B|}{|AUB|}$$

Se deriva del índice de **Jaccard**, el mismo que es conocido para aplicar estadísticas en muestras de conjuntos en intersección sobre unión (Fernández, Velasco & López, 2010), la ecuación de Jaccard se define de la siguiente manera:

$$(a,b) = \frac{|A \cap B|}{|AUB|} = J(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Ecuación 4: Indice de Jaccard
Fuente: (Fernández, Velasco & López 2010)

4.3.3.1 Ejemplos de Cálculo de similitud entre propiedades

Se ha seleccionado varias propiedades de forma aleatoria, para el cálculo de la similitud semántica, entre las ontologías Culture-shop y Library, con el objetivo de detallar su procedimiento y comparar los resultados.

Ontologías Culture-shop y Library

Aplicando el índice de Jaccard entre las dos ontologías, calcular la similitud entre propiedades que son pertenecientes a las siguientes clases:

- Person y Volume
- Product y Human
- Person y Human

El primer paso es calcular el grado de similitud entre las propiedades de las clases Person y Volume, obteniendo la siguiente tabla interna.

Tabla 11: Similitud entre propiedades de las clases Person y Volume

Simi_lexica (C_{oa}, C_{ob})	Title	Year
Firstname	0.33	0.11
Lastname	0.25	0.13

Fuente: El autor
Elaboración: El autor

Una vez que se obtiene la matriz interna del cálculo de grado de similitud entre las propiedades, a estos valores se aplica la ecuación del índice de Jaccard.

El cálculo es el siguiente:

$$J\,(A,B) = \frac{0}{0.33 + 0.25 + 0.11 + 0.13}$$

$$= \frac{0}{0.82}$$

$$= 0$$

Donde 0, es el número de propiedades que tienen una similitud de 1, dividido para la suma de todos los valores obtenidos en la tabla y obteniendo como resultado 0.

El segundo paso es calcular el grado de similitud de las propiedades de las clases Product y Human, mediante la ecuación del índice de Jaccard; obteniendo el resultado de la tabla 12.

Tabla 12: Similitud entre propiedades de las clases Product y Human

Simi_Propiedades (C_{oa}, C_{ob})	Name
Name	1
Id	0
Price	0.2
Topic	0

Fuente: El autor
Elaboración: El autor

$$J\,(A,B) = \frac{1}{1 + 0 + 0.2 + 0}$$

$$= \frac{1}{1.02}$$

$$= 0.833$$

El resultado de este proceso es de 0.833, obtenido a través de 1(propiedades con total similitud), dividido para la suma total de los valores de similitud entre todas las propiedades.

El tercer paso es calcular el grado de similitud entre las propiedades que pertenecen a clases Person y Human, obteniendo la siguiente tabla interna:

Tabla 13: Similitud entre propiedades de las clases Person y Human

Simi_lexica (C_{oa}, C_{ob})	Name
Firstname	0.44
Lastaname	0.44

Fuente: El autor
Elaboración: El autor

$$J\ (A,B) = \frac{0}{0.44 + 0.44}$$

$$= \frac{0}{0.88}$$

$$= 0$$

El resultado final es 0, realizado con los valores las propiedades con similitud 1, en este caso no existen y da un valor de 0, dividido para el número total de las similitudes encontradas.

Luego de realizar las tres tablas internas, se elabora la matriz final de propiedades en base a sus clases, quedando como resultado la tabla 14, similitud final entre propiedades.

Tabla 14: Similitud final entre propiedades

Simi_propiedades (C_{oa}, C_{ob})	Volume	Human
Person	0	0
Product	0	0.833

Fuente: El autor
Elaboración: El autor

EL Anexo 3, presenta la tabla final de similitud entre propiedades, de las ontologías Culture-shop y Library.

La Figura 28, indica el desarrollo de la función programada, para obtener la similitud entre propiedades de las dos ontologías, de acuerdo a los ejemplos de la sección 4.3.3.1, realiza consultas SPARQL para obtener las propiedades existentes, aplica el cálculo del grado de similitud mediante el índice de Jaccard, retornado todos los valores de similitudes.

```
1. function similitudPropiedades($claseOnt1, $claseOnt2, $graph1,
   $graph2) {
2.       $propiedad1 = array();
3.       $propiedad2 = array();
4.
5.       $pref1 = "PREFIX a: <".$graph1.">";
6.       $pref2 = "PREFIX a: <".$graph2.">";
7.
8.       $sparql = $pref1." SELECT * FROM <".$graph1."> WHERE { ?s a
   owl:DatatypeProperty . ?s rdfs:domain a:".$claseOnt1." }";
9.       $result1 = sparql_query( $sparql );
10.            $max_pro1 = sizeof($result1->rows);
11.            for($i = 0; $i < $max_pro1;$i++){
12.                  $pro = generate_atributos($result1-
   >rows[$i]["s"]["value"]);
13.                  array_push($propiedad1, $pro);
14.            }
15.
16.            $sparql = $pref2." SELECT * FROM <".$graph2."> WHERE {
   ?s a owl:DatatypeProperty . ?s rdfs:domain a:".$claseOnt2." }";
17.            $result2 = sparql_query( $sparql );
18.            $max_pro1 = sizeof($result2->rows);
19.            for($i = 0; $i < $max_pro1;$i++){
20.                  $pro = generate_atributos($result2-
   >rows[$i]["s"]["value"]);
21.                  array_push($propiedad2, $pro);
22.            }
23.            $sum = 0; $sumT= 0; $sumAux = 0;
24.            for( $i=0;$i< sizeof($propiedad1);$i++){
25.            for( $j=0;$j< sizeof($propiedad2);$j++){
26.                  $distancia=calcularDistancia($propiedad1[$i],
   $propiedad2[$j]);
27.                  $afinidad=calcularAfinidad($propiedad1[$i],
   $propiedad2[$j], $distancia);
28.
29.                        if($afinidad == 1){
30.                        $sum = $sum + $afinidad;
31.                  }else{
32.                        $sumAux = $sumAux + $afinidad;
33.                  }
34.            }
35.            }
36.            if($sum > 0){
37.                  $sumT = $sum/($sum+$sumAux);
38.            }
39.            return $sumT;
40.
41.      }
```

Figura 28: Función similitudPropiedades
Fuente: El autor
Elaboración: El autor

4.3.4 Similitud entre superclases

Se aplica a partir de dos clases originarias, para poder realizar la similitud entre superclases primero se calcula la similitud léxica y propiedades de las clases, existe una ecuación para poder definir una similitud entre superclases dadas dos clases conocidas como C1 y C2.

Similitud_superclase(C1, C2) = similitud_parcial(C1, C2)

Donde la similitud parcial, es calculada a partir de la siguiente ecuación:

$$Similitud\_parcial(Ca1, Ca2) = \frac{similitud_{léxica(Ca1,Ca2)} + similitud\_propiedad(Ca1,Ca2)}{2}$$

Ecuación 5: Cálculo de similitud entre superclases
Fuente: García, 2015

El objetivo fundamental de la similitud parcial, es extraer un valor que logre abarcar de manera equitativa la similitud léxica y de propiedades, comparando todas las superclases, con el fin de seleccionar la mayor similitud que exista entre ellas, con el propósito de encontrar cuál de las superclases coinciden con clases C1 y C2 (García, 2015).

4.3.4.1 Ejemplo de Cálculo de similitud entre superclases

Ontologías Culture-shop y Library

Se han seleccionado superclases de manera aleatoria, con el propósito de explicar y comparar los resultados obtenidos, Se realiza el cálculo de la similitud semántica entre las superclases de acuerdo a la sección 4.3.4 para las superclases de las siguientes clases.

- Book y Essay

El primer paso de la similitud de las superclases es crear dos tablas internas, en la tabla 15 se indica el primer cálculo mediante el grado de la similitud léxica.

Tabla 15: Similitud superclases de las clases Book y Essay

Simi_lexica (C_{oa}, C_{ob})	volume
Product	0.29

Fuente: El autor
Elaboración: El autor

Donde 0.29, es el valor de distancia que existe entre las dos superclases.

La segunda matriz, es realizada mediante el grado de similitud entre propiedades, que pertenecen a las superclases Product y Volumen, como se muestra en la tabla 16.

Tabla 16: Similitud propiedades de las clases Book y Essay

Simi_propiedades (C_{oa}, C_{ob})	Volumen
Product	0

Fuente: El autor

Elaboración: El autor

En este caso la tabla de similitud entre propiedades tiene un valor de 0, debido a que las superclases Product y Volumen no presentan propiedades.

Para construir la tabla final de las superclases, se seleccionan los valores más altos de la similitud léxica y propiedades, que son aplicados en la ecuación de similitud parcial conforme se detalla en la sección 4.2.4, obteniendo el siguiente proceso.

$$Similitud\_Parcial(Ca1, Ca2) = \frac{0.29 + 0}{2}$$

$$= \frac{0.29}{2}$$

$$= 0.142$$

El resultado obtenido es 0.142, que forma la tabla 17 de la similitud final entre las superclases; el Anexo 5 muestra la tabla completa de las similitudes entre las ontologías Culture-shop y Library.

Tabla 17: Similitud final entre superclases

Simi_propiedades (C_{oa}, C_{ob})	Essay
Book	0.142

Fuente: El autor
Elaboración: El autor

La Figura 29, presenta la función sobre la similitud de superclases, que tiene como ingreso las dos ontologías, se obtiene las superclases mediante consultas SPARQL, realiza el proceso del cálculo de la distancia de similitud léxica y propiedades y se ejecuta la ecuación para obtener el resultado final entre las superclases.

```
1. function similitudSuperClass($claseOnt1, $claseOnt2, $graph1, $graph2) {
2.        $superC1 = array();
3.        $superC2 = array();
4.        $simParcial = array();
5.        $pref1 = "PREFIX a: <".$graph1.">";
6.        $pref2 = "PREFIX a: <".$graph2.">";
7.        $sparql = $pref1." SELECT * FROM <".$graph1."> WHERE {
   a:".$claseOnt1." rdfs:subClassOf ?s }";
8.        $result1 = sparql_query( $sparql );
9.        $max_pro1 = sizeof($result1->rows);
10.       for($i = 0; $i < $max_pro1;$i++){
11.             $posicion_coincidencia = strrpos($result1-
   >rows[$i]["s"]["value"], "nodeID://");
12.             if ($posicion_coincidencia === false) {
13.             $pro = generate_atributos($result1->rows[$i]["s"]["value"]);
14.             array_push($superC1, $pro);
15.       }
16.       }
17.
18.       $sparql = $pref2." SELECT * FROM <".$graph2."> WHERE {
   a:".$claseOnt2." rdfs:subClassOf ?s }";
19.       $result2 = sparql_query( $sparql );
20.       $max_pro1 = sizeof($result2->rows);
21.       for($i = 0; $i < $max_pro1;$i++){
22.             $posicion_coincidencia = strrpos($result2-
   >rows[$i]["s"]["value"], "nodeID://");
23.             if ($posicion_coincidencia === false) {
24.             $pro = generate_atributos($result2->rows[$i]["s"]["value"]);
25.             array_push($superC2, $pro);
26.       }
27.       }
28.       $sum = 0; $sumT= 0; $sumAux = 0; $max1 = sizeof($superC1); $max2 =
   sizeof($superC2);
29.       if($max1 > 0 && $max2 > 0){
30.             for( $i=0;$i< $max1;$i++){
31.             for( $j=0;$j< $max2;$j++){
32.                   $distancia=calcularDistancia($superC1[$i],
   $superC2[$j]);
33.                   $afinidad=calcularAfinidad($superC1[$i],
   $superC2[$j], $distancia);
34.                   $afinidad_pro = similitudPropiedades($superC1[$i],
   $superC2[$j], $graph1, $graph2);
35.
36.                   $simPar = ($afinidad + $afinidad_pro)/2;
37.                   array_push($simParcial, $simPar);
38.             }
39.             }
40.             $val = max($simParcial);
41.       }else{
42.                   $val = 0;
43.             }
44.             return $val;
45.       }
```

Figura 29: Función similitud Superclases
Fuente: El autor
Elaboración: El autor

4.3.5 Clasificación y alineación de valores de similitud con Knn-Vecinos y AdaBoost

Para realizar la clasificación y alineación de valores de similitud, se debe tener una cierta experiencia con criterios o reglas de datos semánticos. La clasificación trabaja en base a la tabla léxica, propiedades y superclases, estos valores son tratados por el algoritmo KNN-Vecinos y AdaBoost, para ser clasificados en categoría de correspondencias fuertes y débiles. Se definen los siguientes parámetros que permiten ejecutar de manera correcta la clasificación de los datos fuertes y débiles.

Umbral de pertenencia: Permite determinar si un elemento de las tablas de similitudes corresponde a una clase.

Umbral: Determina si es que se encuentran valores débiles, para no ser clasificados, este dato es definido en base a las ontologías que se van a trabajar, basándose en la experiencia que tenga el usuario en trabajos de estructuras semánticas.

Número de Iteraciones: Número de iteraciones que utiliza el KNN-Vecinos, recorrido de todos los valores de similitudes, entre más iteraciones existen el resultado es más robusto.

Desviación Inicial: permite definir una nueva clase dependiendo de los datos clasificados.

Rango de Solución: Es un nivel de 0 a 1 para una clase nueva.

Conjunto de prueba: Son los elementos o datos de las similitudes para poder alinear las clases, propiedades o superclases (Uris & Galar, 2015).

Luego de determinar los parámetros, si los valores de las medidas de similitud son erróneos, Knn y AdaBoost no podrá agrupar los datos de las similitudes, ya que se trabaja en base a la información de ingreso; si la información es correcta, el algoritmo empieza a realizar la clasificación de los datos en fuertes y débiles de clases, propiedades y superclases (Zagal, 2008), como se observa en la Figura 30; las correspondencias fuertes son los valores que presentan una similitud mayor o igual al 50 por ciento, lo que determina que se realice de manera correcta la alineación de las ontologías; la siguiente clasificación es para las correspondencias débiles de las clases, propiedades y superclases, se forman por los datos que no cumplen con un cierto grado similitud semántica, su promedio es menor al 50 por ciento, siendo separados para el proceso de integración.

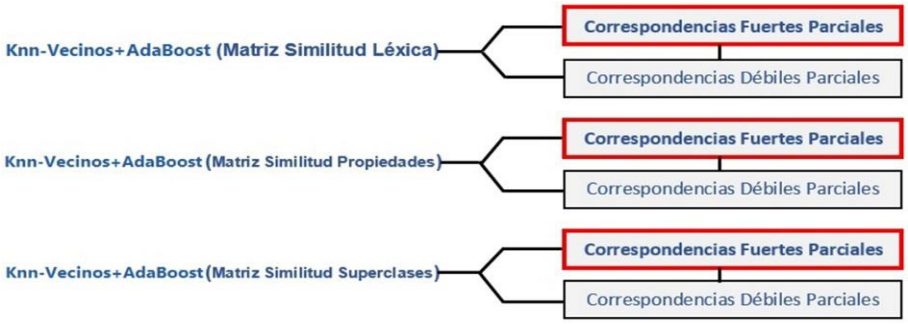

Figura 30: Clasificación de Knn-Vecinos y AdaBoost
Fuente: (Aguilar, 2015)
Elaboración: (Aguilar, 2015)

La Figura 31, indica todo el proceso de recorrido que realiza el algoritmo Knn-Vecinos y AdaBoost, tiene como entrada las medidas de similitud, se genera un arreglo de dos posiciones, en la primera posición se guarda el valor máximo ingresado por las afinidades y en la segunda el valor de la desviación estándar, en base a la experiencia con estructuras semánticas, todo esto se recorre de acuerdo a las iteraciones y a los valores que se van a evaluar; de cada dato evaluado se resta el valor de la posición correspondiente a la afinidad, determinando el resultado máximo, todo esto es elevado al cuadrado para el valor de la variable en la segunda posición, que es la desviación inicial, el resultado obtenido se lo conoce como distancia, donde 1 es para los valores fuertes y 0 para los débiles; estos se clasifican de acuerdo a los parámetros de correspondencia que han sido explicados, obteniendo como resultado final la correspondencia de semántica fuerte, conocidos como alineación de ontologías

```
1.       $D = K_Vecinos($Matriz_similitud,
   $umbral_pertenencia,$desviacion_ini, $rango_solucion,
   $iteraciones,$umbral, $num_clases1, $num_clases2);
2.       $datos2 = array();
3.       $F = $D[0];
4.       $d = $D[1];
5.       $distEucla = $D[2];
6.       $cont=0;$i=0;
7.       for ($i=0; $i < sizeof($Matriz_similitud); $i++) {
8.             foreach ($d as &$debiles) {
9.                   if($i==$debiles)
10.      $datos2[$cont]=$Matriz_similitud[$i];
11.                            $cont++;
12.                            break;
13.                      }
14.                }
15.           }
16.           $desviacion_ini = 1*0.5*$iteraciones;
17.           $De = K_Vecinos($Matriz_similitud,
   $umbral_pertenencia,$desviacion_ini, $rango_solucion,
   $iteraciones,$umbral, $num_clases1, $num_clases2);
18.           $D = $De[0];
19.           return $Analisis = [$F,$D,$distEucla]
20.           $distEuclaFuerte = array();
21.                      for ($h=0; $h < $iteraciones; $h++) {
22.                            for ($i=0; $i <
   sizeof($Matriz_similitud); $i++) {
23.                                  for ($j=0; $j < sizeof($C);
   $j++) {
24.                                        $resta
   =$Matriz_similitud[$i]-$C[$j][0];
25.                                        $potencia = pow($resta,
   2);
26.                                        if($C[$j][1] != 0){
27.                                              $dist =
```

```
28.      $potencia/$C[$j][1];
29.                                                  if($dist <
   $umbral_pertenencia) {
30.                                             $p[$i]=1;
31.                                             $distEuclaFuerte[$i]=
   $dist;
32.      }else{
33.                                                        $p[$i]=0;
34.      }
35.                                       }else{
36.                                             $p[$i]=0;
37.                                       }
38.                                       }
39.                                  }
40.            }
41.      if($D != null){
42.                  $datos2 = array();
43.                  //if($D != 0){
44.                  $F = $D[0];
45.                  $d = $D[1];
46.                  $distEucla = $D[2];
47.
48.                  $cont=0;$i=0;
49.
50.                  for ($i=0; $i < sizeof($Matriz_similitud); $i++)
   {
51.                        foreach ($d as &$debiles) {
52.                              if($i==$debiles){
53.
         $datos2[$cont]=$Matriz_similitud[$i];
54.                                    $cont++;
55.                                    break;
56.                              }
57.                        }
58.                  }
59.                  $desviacion_ini = 1*0.5*$iteraciones;
60.
61.                  $De = K_Vecinos($Matriz_similitud,
   $umbral_pertenencia,$desviacion_ini, $rango_solucion,
   $iteraciones,$umbral, $num_clases1, $num_clases2);
62.                  $D = $De[0];
63.
64.                  return $Analisis = [$F,$D,$distEucla];
65.            }else{
66.                  return 0;
67.            }
```

Figura 31: Distancia Euclidiana
Fuente: El autor
Elaboración: El autor

La Figura 32, muestra el método que identifica los valores que no fueron clasificados por Knn-Vecinos y AdaBoost.

```
68.      Obtener elementos o patrones mal clasificados:
69.      conjunto_prueba=Obtener_no_clasificados
   (Kparametros.conjunto_prueba, D);
70.      Si (Conjunto_prueba está vacío) {
71.      Imprimir: "Todos los elementos clasificados"
72.      Salir de AdaBoost
73.      }
```

Figura 32: Valores no clasificados por Knn-Vecinos y AdaBoost
Fuente: Uris & Galar, 2015
Elaboración: Uris & Galar, 2015

La tabla 18, indica el objetivo logrado en el proceso de alineación entre ontologías.

Tabla 18: Resumen del proceso de alineación de ontologías

Proceso de Alineación	Detalle
Datos de ingreso	Dos ontologías.
Detalle	Ingreso y proceso de las medidas de similitud.
Resultados obtenidos	Tabla de correspondencia de semántica fuerte (Alineación de ontologías).

Fuente: El autor
Elaboración: El autor

4.3.6 Implementación de Enlazado de Ontologías

Para desarrollar la técnica del enlazado, es requisito que las dos ontologías se encuentren alineadas, se utiliza la tabla de correspondencia semántica fuertes como datos de entrada, el algoritmo busca y selecciona el valor máximo de toda la tabla de alineación, para obtener el porcentaje mínimo de similitud, el mismo que sirve para determinar que todos los valores que se encuentren igual o sobre ese porcentaje pasan a ser datos equivalentes y enlazados, recorre toda la alineación, ordena y presenta todos los conceptos que cumplen con la equivalencia.

La Figura 33, determina el porcentaje mínimo a utilizar, definido por el usuario, se selecciona el valor máximo y el valor mínimo es igual al porcentaje mínimo por el valor máximo dividido para 100, obteniendo como resultado los valores mayores al porcentaje determinado por el usuario.

```
1. $pormin = 30;
2. $valMax = max($al_clase[2]);
3. $valorMin = ($pormin*$valMax)/100;
4. $i=0;$i=0
```

Figura 33: Porcentajes Enlazado
Fuente: El autor
Elaboración: El autor

La Figura 34, muestra la función que se encarga de realizar el proceso de enlazado completo entre dos ontologías, determina los porcentajes, consultas SPARQL sobre la tabla de alineación, obteniendo como resultado final una tabla de todos los conceptos que son equivalentes o enlazados.

```
1. foreach ($al_clase[2] as $value) {
2.       if($value>$valorMin){
3.             $en_clases1[$j] = $aclass1[0][$i];
4.             $en_clases2[$j] = $aclass1[1][$i];
5.             $j++;
6.       } $i++; }
7. $result["porcentEnl"] = $pormin;
8. $result["numEnl"] = sizeof($en_clases1);
9. $result["clase_en_Ont1"] = $en_clases1;
10.      $result["clase_en_Ont2"] = $en_clases2;
11.      $claseC = $clase1;
12.      //alineacion
13.      $aliOntclass1 = $aclass1[0];
14.      $aliOntclass2 = $aclass1[1];
15.      $aliOntvalor2 = $al_clase[2];
16.            $padre = array();
17.            $subClass = array();$contA = 0;
18.      for ($i=0; $i < sizeof($aliOntclass2); $i++) {
19.            $pref1 = "PREFIX a: <".$graph2.">";
20.            $sparql = $pref1." SELECT * FROM <".$graph2."> WHERE {
   ?s rdfs:subClassOf a:".$aliOntclass2[$i]." . }";
21.            $result1 = sparql_query( $sparql );
22.            $max_pro1 = sizeof($result1->rows);
23.            $cont = 0;
24.            if($max_pro1 > 0){
25.                  while ($cont < $max_pro1) {
26.                        $subclase = generate_atributos($result1-
   >rows[$cont]["s"]["value"]);
27.                        if (!in_array($subclase, $aliOntclass2)) {
28.                              $padre[$contA] = "Thing";
29.                              $subClass[$contA] = $subclase;
30.                        }else{
31.                              $padre[$contA] = "Thing";
32.                              $subClass[$contA] =
   $aliOntclass2[$i];
33.                              $contA++;
34.                              $padre[$contA] = $aliOntclass2[$i];
35.                              $subClass[$contA] = $subclase;
36.                        }
37.                        $cont++;
38.                        $contA++;
39.                  }
40.            }
41.
42.      }
```

Figura 34: Enlazado general de ontologías
Fuente: El autor
Elaboración: El autor

La tabla 19, muestra el resultado del objetivo alcanzado en el proceso del enlazado de ontologías.

Tabla 19: Resumen del proceso de enlazado de ontologías

Proceso de Enlazado	Detalle
Datos de ingreso	Dos ontologías alineadas.
Detalle	Se crea una tabla con los nodos enlazados, y un grafo final de una nueva ontología a través de los conceptos que son equivalentes, dejando por fuera el conocimiento.
Resultados obtenidos	Tabla y grafo del enlazado de los conceptos equivalentes.

Fuente: El autor
Elaboración: El autor

4.3.7 Implementación de Mezclado de Ontologías

La técnica de mezclado, requiere como ingreso obligatorio que las ontologías se encuentren alineadas, para el desarrollo se requiere de trabajar con dos algoritmos conocidos como mezcla débil y mezcla fuerte, como entrada utilizan la tabla final de correspondencia de semántica fuerte, al momento de aplicar el algoritmo débil existe el riesgo de que no exista un mezclado completo de los conceptos, dejando el conocimiento por fuera, mientras que el algoritmo fuerte, se encarga de verificar si todos los conceptos fueron agregados por la primera mezcla débil, en caso de que no hayan sido incorporados, va integrando todos los nodos sueltos en el universo si el caso lo amerita, o en las clase padre donde debe ir el mezclado correcto, obteniendo las relaciones; ejecuta otro recorrido en base a los subárboles, aplica la integración de todos los nodos en las clases padre a las que pertenezcan, logrando incluir todo el conocimiento, con el objetivo de ir armando una estructura semántica bastante robusta, denominada ontología C, compuesta por A y B.

El mezclado de ontologías, presenta un significado muy fuerte por que logra incluir todo el conocimiento, enriqueciendo por completo las estructuras semánticas.

La Figura 35, indica la función del mezclado, como entrada recibe la tabla de correspondencias semánticas fuertes de las dos ontologías, ejecuta una mezcla débil, que genera una nueva clase (ClaseC), la cual se compone de las clases de la primera ontología, luego recorre las clases de la segunda ontología, obteniendo las subclases con dos resultados; si el primer resultado de las subclases no se encuentra en la tabla de alineación, esta toma como padre al espacio o a Thing, en el caso de que los datos si se encuentren en la tabla de alineamiento se copia toda la relación, es decir, se toma a Thing como padre, luego la subclase de este corresponde a la clase alineada de la ontología; después se adiciona una siguiente posición para una nueva alineación de clases, finalmente se realiza

todo el recorrido de las clases y se obtiene todas las relaciones, terminando la mezcla débil y retomando la mezcla fuerte que obtiene todos los datos que no fueron copiados ni alineados, es decir que no se encuentren como subclases o padres, si existe algún concepto que no fue copiado, realiza el proceso de la mezcla débil copiando todo el subárbol al espacio, luego se crea la relación con el ancestro y el universo, se copia todas las relaciones entre los conceptos del subárbol no integrados; buscando las subclases que se encuentren alineadas para ser relacionadas, este proceso se repite para todas las clases que no se encuentran alineadas ni realizadas por la mezcla débil, finalmente la ClaseC realiza un recorrido que verifica si los datos se encuentran en la tabla de correspondencias fuertes, si los mismos no se encuentran en la tabla toma como padre al espacio o a Thing; de lo contrario si los conceptos se encuentran en las correspondencias, se relaciona el padre que sería la clase alineada con la subclase, terminando todo el recorrido y obteniendo los padres junto con su subclases.

```
1. /*Algoritmo mezcla debil*/
2. $claseC = $clase1;
3. //alineacion
4. $aliOntclass1 = $aclass1[0];
5. $aliOntclass2 = $aclass1[1];
6. $aliOntvalor2 = $al_clase[2];
7.        $padre = array();
8.        $subClass = array();$contA = 0;
9. for ($i=0; $i < sizeof($aliOntclass2); $i++) {
10.       $pref1 = "PREFIX a: <".$graph2.">";
11.
12.       $sparql = $pref1." SELECT * FROM <".$graph2."> WHERE { ?s
   rdfs:subClassOf a:".$aliOntclass2[$i]." . }";
13.       $result1 = sparql_query( $sparql );
14.       $max_pro1 = sizeof($result1->rows);
15.       $cont = 0;
16.       if($max_pro1 > 0){
17.             while ($cont < $max_pro1) {
18.                   $subclase = generate_atributos($result1-
   >rows[$cont]["s"]["value"]);
19.                   if (!in_array($subclase, $aliOntclass2)) {
20.                         $padre[$contA] = "Thing";
21.                         $subClass[$contA] = $subclase;
22.                   }else{
23.                         $padre[$contA] = "Thing";
24.                         $subClass[$contA] = $aliOntclass2[$i];
25.                         $contA++;
26.                         $padre[$contA] = $aliOntclass2[$i];
27.                         $subClass[$contA] = $subclase;
28.                   }
29.                    $cont++;
30.                   $contA++;
31.             }   }  }
32. /*FIN Algoritmo mezcla debil*/
```

```
33. $CnoCA = array();$contA=0;
34. for ($i=0; $i < sizeof($clase2); $i++) {
35.       $bandera = 1;
36.       for ($j=0; $j < sizeof($subClass); $j++) {
37.             if($clase2[$i] == $subClass[$j]){
38.                   $bandera = 0;
39.                   break;
40.             }
41.       }
42.       for ($j=0; $j < sizeof($padre); $j++) {
43.             if($clase2[$i] == $padre[$j]){
44.                   $bandera = 0;
45.                   break;
46.             }
47.       }
48.       if($bandera == 1){
49.             $CnoCA[$contA] = $clase2[$i];
50.             $contA++;
51.       }
52. }
53. if(sizeof($CnoCA) > 1){
54.       $contA=sizeof($subClass);
55.       for ($i=0; $i < sizeof($CnoCA); $i++) {
56.             $pref1 = "PREFIX a: <".$graph2.">";
57.             $sparql = $pref1." SELECT * FROM <".$graph2."> WHERE {
    a:".$CnoCA[$i]." rdfs:subClassOf ?s }";
58.             $result1 = sparql_query( $sparql );
59.             $max_pro1 = sizeof($result1->rows);
60.             $cont = 0;
61.             if($max_pro1 > 0){
62.                   while ($cont < $max_pro1) {
63.                         if($result1->rows[$cont]["s"]["type"] ==
    "uri"){
64.                               $superClass =
    generate_atributos($result1->rows[$cont]["s"]["value"]);
65.                               if(!is_null($superClass)){
66.                                     $padre[$contA] = "Thing";
67.                                     $subClass[$contA] = $superClass;
68.                                     $contA++;
69.                                     $padre[$contA] = $superClass;
70.                                     $subClass[$contA] = $CnoCA[$i];
71.                                     $contA++;
72. $pref1 = "PREFIX a: <".$graph2.">";
73.
74.                                     $sparql = $pref1." SELECT * FROM
    <".$graph2."> WHERE { ?s rdfs:subClassOf a:".$CnoCA[$i]." . }";
75.                                     $result1 = sparql_query( $sparql
    );
76.                                     $max_pro1 = sizeof($result1-
    >rows);
77.                                     $cont = 0;
78.                                     if($max_pro1 > 0){
79.
```

```
80. while ($cont < $max_pro1) {
81.                                                     $subclase =
    generate_atributos($result1->rows[$cont]["s"]["value"]);
82.                                                     if
    (in_array($subclase, $aliOntclass2)) {
83.                                                             $padre[$contA]
    = $aliOntclass2[$i];
84.
          $subClass[$contA] = $subclase;
85.                                                     }
86.                                                     $cont++;
87.                                                     $contA++;
88.                                                 }
89.                                             }
90.                                         }
91.                             }
92.                             $cont++;
93.                     }
94.             }
95.       }
96. }
97.
98.
99. $contA=sizeof($subClass);
100.      for ($i=0; $i < sizeof($claseC); $i++) {
101.            $pref1 = "PREFIX a: <".$graph1.">";
102.
103.            $sparql = $pref1." SELECT * FROM <".$graph1."> WHERE { ?s
    rdfs:subClassOf a:".$claseC[$i]." . }";
104.            $result1 = sparql_query( $sparql );
105.            $max_pro1 = sizeof($result1->rows);
106.            $cont = 0;
107.            if($max_pro1 > 0){
108.                  while ($cont < $max_pro1) {
109.                        $subclase = generate_atributos($result1-
    >rows[$cont]["s"]["value"]);
110.                        if (!in_array($subclase, $aliOntclass1)) {
111.                              $padre[$contA] = "Thing";
112.                              $subClass[$contA] = $subclase;
113.                        }else{
114.                              $padre[$contA] = $aliOntclass2[$i];
115.                              $subClass[$contA] = $subclase;
116.                        }
117.                        $cont++;
118.                        $contA++;
119.                  }
120.            }
121.
122.      }
123.      //array_multisort($padre, $subClass);
124.      $result["padre"] = $padre;
125.      $result["subClass"] = $subClass;
```

Figura 35: Mezclado fuerte de ontologías
Fuente: El autor
Elaboración: El autor

Se describe el objetivo logrado en el proceso del mezclado de ontologías, como se indica en la tabla 20

Tabla 20: Resumen del proceso de mezclado de ontologías

Proceso de Mezclado	**Detalle**
Datos de ingreso	Dos ontologías alineadas.
Detalle	El mezclado integra todo el conocimiento, realiza una mezcla débil y fuerte, crea un grafo final de una nueva ontología a través de todos los conceptos mezclados, logrando incluir todo el conocimiento, que fue dejado por fuera en la técnica del enlazado.
Resultados obtenidos	Grafo de una nueva ontología, con todo el conocimiento integrado.

Fuente: El autor
Elaboración: El autor

CAPITULO V ANÁLISIS DE RESULTADOS

En el presente capítulo se realiza el análisis de las tres técnicas obtenidas de minería ontológica; indicando las diferencias que existen entre las mismas y cuál es la técnica más óptima y recomendable a utilizar.

5.1 Resultado de la Técnica de Alineación de Ontologías

La alineación de ontologías, cumple un papel fundamental encontrando todas las correspondencias semánticas, realiza un proceso completo conforme lo estudiado en el capítulo 4, requiere un sinnúmero de entradas para poder ser aplicado, de acuerdo a las estructuras semánticas que serán analizadas los tiempos de ejecución tendrán variación, se alinean clases, propiedades y superclases en base a las tablas de similitud calculadas, en este trabajo se desarrolla una aplicación Web que permite analizar cualquier tipo de ontologías. Se ha seleccionado Culture-shop y Lybrary, de acuerdo a su estructura se determina un umbral con un valor de 15, que sirve como referencia para cada cálculo que corresponde a todas las afinidades obtenidas, se estable un umbral de pertenencia 0.1, que sirve para evaluar que los valores de distancia sean menores a este umbral, determinando 1 para los datos fuertes y 0 para los débiles. Estos umbrales son definidos de acuerdo a la estructura de las ontologías y experiencia que se tenga trabajando en datos semánticos.

La tabla 21, indica claramente 2 ontologías alineadas, en la parte izquierda se encuentra Culture-shop y en la derecha Lybrary, presentan como resultado de alineación 51 parejas de clases, con una correspondencia semántica muy fuerte, 0 propiedades con la técnica de alineado, debido a que la estructura original tiene menos propiedades y al momento de aplicar el proceso del alineado presenta como resultado una correspondencia muy baja, finalmente se alinean 6 superclases, en base a las 7 originales que conforman estas ontologías, lo que hace que exista una correspondencia fuerte. Este proceso ha tenido un tiempo de ejecución de 8 minutos, debido a que la estructura de las ontologías no demasiado extenso, obteniendo como resultado final una técnica de alineación robusta y exitosa de todos los conceptos encontrados en las 2 estructuras semánticas originales.

Es muy importante utilizar el proceso de alineación, porque la mayoría de los conceptos originales son formados y presentados en un nuevo resultado óptimo y robusto, que sirve como base principal para poder aplicar la técnica de enlazado y mezclado.

Tabla 21: Alineación final de ontologías Culture-shop y Library

TIPO DE SIMILITUD	PAREJAS	
Similitud Léxica	Book	Autobiography
	Book	Biography
	Book	Novel
	Book	Poetry
	Book	Politics
	Book	Volume
	Children	Autobiography
	Children	Biography
	Children	Human
	Children	Literature
	Children	Novel
	Children	Poetry
	Person	Biography
	Person	Essay
	Person	Human
	Person	LiteraryCritic
	Person	Literature
	Person	Poetry
	Person	Politics
	Pocket	Biography
	Pocket	Literature
	Pocket	Volume
	Pocket	Writer
	Popular	Autobiography
	Popular	Biography
	Popular	Essay
	Popular	LiteraryCritic
	Popular	Literature
	Popular	Writer
	Product	Biography
	Product	LiteraryCritic
	Product	Literature
	Product	Novel
	Product	Writer
	Publisher	Autobiography
	Publisher	Essay
	Publisher	Human
	Publisher	Literature
	Publisher	Novel
	Publisher	Poetry
	Publisher	Volume

TIPO DE SIMILITUD	PAREJAS	
Similitud Léxica	Science	Human
	Science	Literature
	Science	Novel
	Science	Poetry
	Science	Volume
	Science	Writer
	Textbook	Autobiography
	Textbook	Literature
	Textbook	Poetry
	Textbook	Writer
Similitud entre propiedades		
Similitud entre superclases	Book	Essay
	Book	Literature
	CD	Essay
	CD	Literature
	DVD	Essay
	DVD	Literature

Fuente: El autor
Elaboración: El autor

5.2 Resultado de la Técnica de Enlazado de Ontologías

El enlazado es otra de las técnicas de la minería ontológica, se obtiene a través de los valores alineados, para aplicar esta técnica se selecciona dos estructuras semánticas que son Culture-shop y Library, en donde se empieza determinando un porcentaje mínimo, con un valor equivalente a 30, que permite obtener todos los valores que se van a enlazar; es decir todos los conceptos alineados que presenten un resultado mayor al porcentaje mínimo serán los que cumplen la condición para enlazarse.

La tabla 22, muestra un resultado completo de todo el enlazado de ontologías, indica en la posición izquierda todos los conceptos de Culture-shop que son equivalentes, con los nodos de la parte derecha de Library, presentando como resultado 15 parejas de conceptos enlazados.

Tabla 22: Enlazado de ontologías

TIPO DE SIMILITUD	PAREJAS	
Numero de enlaces 15 con porcentaje mínimo de similitud 50	Book	Biography
	Children	Biography
	Children	Literature
	Person	Biography
	Pocket	Biography
	Pocket	Literature

TIPO DE SIMILITUD	PAREJAS	
	Popular	Biography
	Product	Biography
	Publisher	Essay
	Publisher	Human
	Publisher	Literature
	Publisher	Novel
	Publisher	Poetry
	Publisher	Volume
	Textbook	Literature

Fuente: El autor
Elaboración: El autor

La Figura 36, muestra el grafo de las 2 ontologías originales que cumplirán el proceso de enlazado.

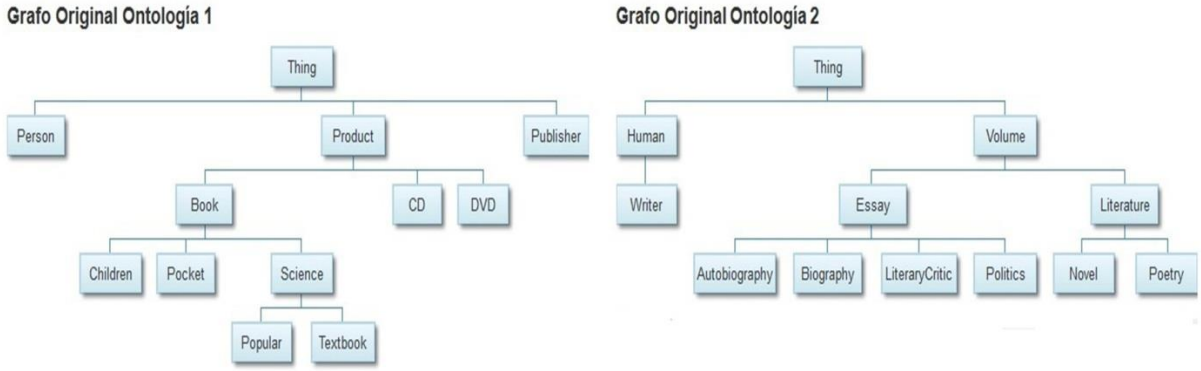

Figura 36: Grafos originales para el enlazado de ontologías Culture-shop y Library
Fuente: El autor
Elaboración: El autor

La Figura 37, presenta un nuevo grafo, aplicando el enlazado de las ontologías Culture-shop y Library, se puede observar todas las clases padre, que contienen sus conceptos enlazados de acuerdo a sus equivalencias, su tiempo de ejecución es de 5 minutos, es muy corto porque solo va seleccionando todos los conceptos equivalentes sin importar no incluir todo el conocimiento.

Esta técnica no lograr incluir todos los conceptos, dejando por fuera el conocimiento lo que hace que el resultado final no sea extenso y robusto, presentando ciertas inconsistencia en los conceptos que no son equivalentes, los mismos que no son tomados en cuenta por esta técnica al momento de enriquecer las ontologías. Dando como resultado una tercera ontología enlazada solo con las equivalencias, la misma que no presenta una total estructura enriquecida.

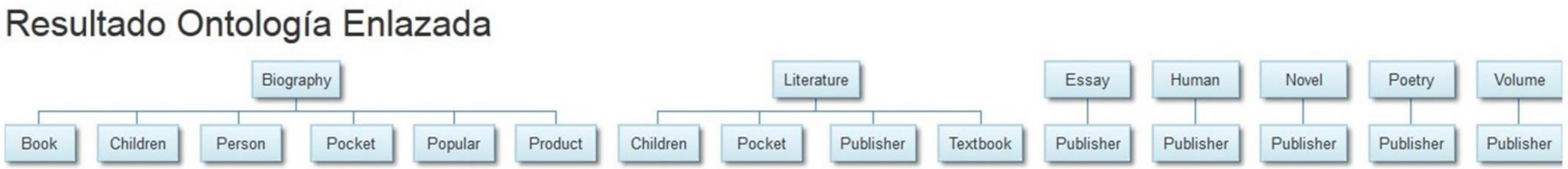

Figura 37: Grafo final de enlazado de ontologías Culture-shop y Library
Fuente: El autor
Elaboración: El autor

5.3 Resultados de la Técnica de Mezclado de Ontologías

Se utiliza la tabla de correspondencias fuertes de la alineación entre Culture-shop y Library para el mezclado de las ontologías, su objetivo principal es de lograr un nuevo resultado con todos los conceptos integrados, es decir agregando por completo el conocimiento, la Figura 38 muestra el grafo de las 2 ontologías origen que serán mezcladas.

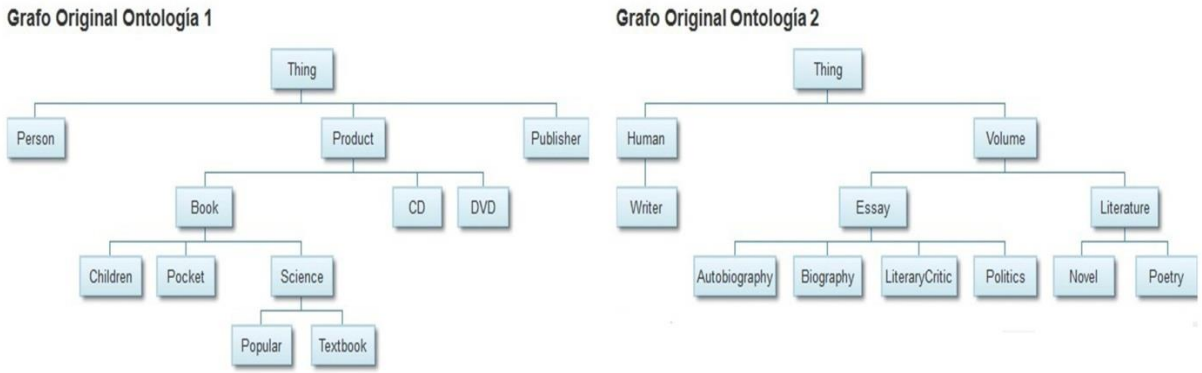

Figura 38: Grafos originales para el mezclado de ontologías Culture-shop y Library
Fuente: El autor
Elaboración: El autor

La Figura 39, indica el grafo final del mezclado de ontologías, se puede observar que existe una sola clase padre Thing, en donde todos los conceptos han sido mezclados de manera correcta, incluyendo todo el conocimiento que fue dejado por fuera en la técnica del enlazado, esto significa que no se pierde ningún nodo, presentando como resultado una nueva ontología completa y robusta, su tiempo de ejecución es de 10 minutos el doble de la ejecución en el enlazado, porque aquí se realiza la integración de los conceptos sin dejar nada por fuera.

La técnica del mezclado es recomendada en la minería ontológica, porque se compone de una mezcla débil y su estructura es muy robusta, totalmente diferente al enlazado, en donde todo el conocimiento se encuentra integrado, enriqueciendo por completo las estructuras semánticas.

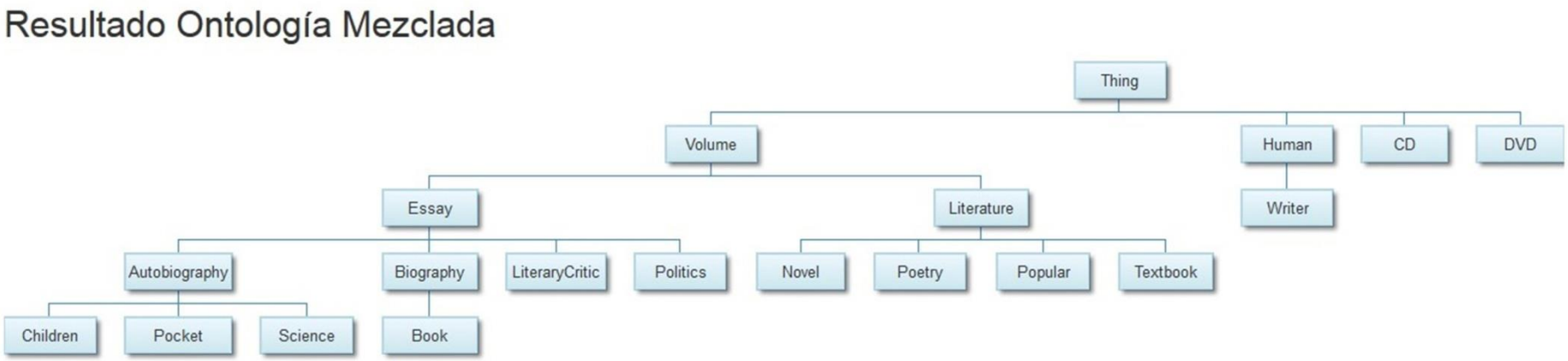

Figura 39: Grafo final de mezclado de ontologías Culture-shop y Library
Fuente: El autor
Elaboración: El autor

Se concluye que la técnica recomendada para utilizar en la minería ontológica, es el mezclado de ontologías, porque esta técnica permite generar una nueva estructura semántica de forma completa y robusta, diferente al enlazado de las ontologías que no incorpora todo el conocimiento, es decir solo enlaza los datos o conceptos equivalentes. El mezclado tiene la capacidad de abarcar y unir todo el conocimiento, es decir enriqueciendo por completo cualquier tipo de estructuras semánticas que cumplan con el proceso de análisis y aplicación de la minería ontológica.

TRABAJOS FUTUROS

La elaboración de este trabajo sirve para desarrollar investigaciones futuras y aplicaciones, a continuación se describen algunas ideas que pueden ser implementadas.

- Implementar el desarrollo del archivo owl para la técnica de alineación, enlazado y mezclado de ontologías, con su estructura semántica completa, con el objetivo de generar archivos descargables por cada análisis realizado en las técnicas de minería ontológica.
- Automatizar por completo el proceso de umbrales para los algoritmos de minería ontológica, lo cual permitirá que el usuario final ejecute las técnicas con o sin experiencia en datos semánticos.

CONCLUSIONES

Una vez que se ha concluido todas las fases del presente trabajo de titulación, se presenta las conclusiones en base a los objetivos planteados para este proyecto.

- Las medidas de similitud, cuentan con un completo proceso matemático, lo cual nos permiten obtener un complemento necesario, que permite determinar todos los datos que serán base para poder implementar un correcto alineamiento.
- Los procesos de alineación son basados en medidas de similitud, en este caso se ha seleccionado el proceso adecuado para obtener una correcta correspondencia semántica.
- Se ha concluido que el enlazado de ontologías, solo permite emparejar los conceptos que son equivalentes dejando todo el conocimiento por fuera, y a su vez se lo determina como una técnica no robusta, con varias inconsistencias.
- En este caso se concluye de manera diferente que el mezclado de ontologías, es mucho mejor en comparación al enlazado, ya que por su robustez, incluye una mezcla débil y fuerte, que logra unir todo el conocimiento que es dejado por fuera en el enlazado, con el propósito de integrar todos los conceptos, enriqueciendo de manera completa todas las estructuras semánticas.
- Seguidamente se implementó una aplicación Web, que permite analizar cualquier tipo de estructuras semánticas, presentando como resultados los grafos de las ontologías origen, con la alineación, enlazado y mezclado de las mismas.
- La finalidad del proyecto es crear todas las integraciones o incorporaciones de los conceptos de manera automática, entre varias ontologías con un correcto funcionamiento.
- Se ha logrado comparar la alineación, enlazado y mezclado de estructuras semánticas, llegando a determinar que la técnica más recomendable que logra incluir todo el conocimiento es el mezclado, cumpliendo el objetivo de enriquecer las ontologías.
- Se logra concluir que los algoritmos de la minería ontológica utilizados en este proyecto, no dependen de ninguna base de datos, por su estructura son de fácil compresión, lo cual ha permitido descubrir las correspondencias de la semántica de forma completa.

RECOMENDACIONES

Con los resultados obtenidos en el presente trabajo de titulación, se elabora las siguientes recomendaciones con el fin de dar continuidad a trabajos futuros sobre este campo investigativo.

- Se debe tratar de optimizar los tiempos de ejecución de los 3 algoritmos de la minería ontológica, con la finalidad de reducir los tiempos actuales.
- Se sugiere investigar una nueva técnica de minería ontológica, la cual permita incorporar de todo el conocimiento, todo esto sin la necesidad de determinar un umbral en base a la experiencia en datos semánticos.
- Aplicar las técnicas de la minería ontológica, con estructuras semánticas que sean de diferentes dominios.
- Se recomienda que la aplicación Web desarrollada en este proyecto siga adaptándose a nuevas tecnologías futuras, con el propósito de que la aplicación continúe trabajando con datos semánticos, con el objetivo de seguir enriqueciendo la minería ontológica, presentando una interfaz de fácil uso para el usuario final.
- Seguir investigando todos los servicios de la minería ontológica, con el propósito de trabajar con estructuras de datos semánticos en cualquier tipo de archivos, sin necesidad que sean de extensión owl.

BIBLIOGRAFIA

- Aguilar, Jose, E. De, Ingeniería, S. F. de, Andes, U. de L., & Mérida, V. (2015). Big Data , Minería avanzada : minería semántica , minería de datos.
- Berners-Lee. (2001). Web Semántica - XML2000 - slide "Arquitectura" Retrieved April 3, 2017, from https://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html
- Caliusco, M. L. & Stegmayer, G. (2010). *Emergent Web Intelligence: Advanced Semantic Technologies*. *Emergent Web Intelligence: Advanced Semantic Technologies*. https://doi.org/10.1007/978-1-84996-077-9
- Cantador Ivan (2005). Aplicación de Perceptrones Paralelos y AdaBoost a Problemas de Clasificación Desequilibrados.
- Censor Cosmico. (2016). Código Java: Cálculo del grado de similitud entre dos palabras. Distancia de Levenshtein. Retrieved July 1, 2017, from https://censorcosmico.blogspot.com/2016/11/calculo-del-grado-de-similitud-o.html
- Chavez Rodrigo Cornejo (2006). Disponible en: http://www.redalyc.org/articulo.oa?id=55140109. *Experimento Educativo*, 13.
- De Diego, R., Fernández López, M., Gómez-Pérez, A., & Ramos, J. A. (2005). OEGMerge: un modelo de mezcla de ontologías basado en casuísticas. *XI Conferencia de La Asociación Española Para La Inteligencia Artificial 2005*, *I*(November), 283–292. https://doi.org/ISBN: 84-96474-13-5
- Doctoral, T. (2014). Minería de Datos en Análisis Ontológico-Funcionales.
- Euzenat, Jérome, & Shvaiko, P. (2013). *Ontology Matching*. *Evaluation, Second Edition*.
- Fernández-caballero, (2016). Ontologías del modelo del alumno y del modelo del dominio en sistemas de aprendizaje adaptativos y colaborativos, (May 2005).
- Fernández-Hernández, Carbonell-de-la-Fe, Pérez-González & Villalón-Aguilera, T. (2009). Las ontologías nuevos retos. *Isko-Spain*, 355–379.
- Fernández, Susel, & Pértegas, S. (2002). Investigación cuantitativa y cualitativa. *Cadena Atención Primaria*, (Figura 1), 76–78. Retrieved from http://www.fisterra.com/mbe/investiga/cuanti_cuali/cuanti_cuali.asp
- Fernández Susel, Velasco Juan R. & López-Carmona, M. A. (2010). Sistema basado en reglas difusas para el mapeo de ontolog'. *Uhues*, 363–368. Retrieved from http://www.uhu.es/estylf2010/trabajos/SS04-03.pdf
- García Neili Machado (2015). Uso de la similitud semántica para la recuperación de información geoespacial .

- Geospatial Semantic Web Community Group. (2015). Retrieved March 6, 2017, from https://www.w3.org/community/geosemweb/
- Gómez Ruiz. (2013). Aplicación bibliográfica usando Linked Data. Retrieved from https://eciencia.urjc.es/handle/10115/12103.
- Gonzáles Carlos, Rodriguez Juan José (2010). Aprendizaje Automático y Contenidos
- Mikel Xabier Uriz Martin & Mikel Galar Idoate (2015). Aprendizaje de distancias basadas en disimilitudes para el algoritmo de clasificación kNN.
- Guarino Nicola. (2006). Formal Ontology and Information Systems. *Formal Ontology and Information Systems*, 13. Retrieved from http://www.loa.istc.cnr.it/old/Papers/FOIS98.pdf
- Jose Miguel Gascueña, Antonio Fernandez, P. G. (2005). Ontologías del modelo del alumno y del modelo del dominio en sistemas de aprendizaje adaptativos y colaborativos. *User Modeling- Springer Berlin Heidelberg*, (May 2005), 367–376. Retrieved from http://www.aipo.es/
- Lamarca Lapuente María Jesús. (2013, December). Hipertexto, el nuevo concepto de documento en la cultura de la imagen. https://doi.org/10.1109/5254.920597
- Likavec Silvia, Osborne Francesco, & Cena Federica (2015). Property-based Semantic Similarity and Relatedness for Improving Recommendation Accuracy and Diversity. *International Journal on Semantic Web and Information Systems*, *11*(4), 1–40. https://doi.org/10.4018/IJSWIS.2015100101.
- Minería de datos: área de oportunidades, área de procesos. (2012).
- Nigro Hector Oscar & Císaro Sandra Gonzáles (2006). El Proceso de Minería de Datos Asistido por Ontologías, (2004).
- Open, H. by the K. F. (2016). Linked Open Vocabularies. Retrieved from https://lov.okfn.org/dataset/lov/vocabs/foaf
- Ossowski, Sascha (2014). Ontologías y Web Semántica, año academico 2014 & 2015
- Rangel Piña, Carlos Ramon (2015). Integración de Ontologías desde el punto de vista de Minería Ontológica y del Paradigma de Arquitecturas Orientadas a Servicios. *Escuela de Ingeniería de Sistemas*, 1–129. https://doi.org/10.1007/s13398-014-0173-7.2
- Suero Daniel Vila (2012). Datos Enlazados en la BNE: datos.bne.es
- Valencia Zapata, Gustavo Adolfo (2008). Minería de datos como herramienta para toma de decisiones estratégias, 1–18. Retrieved from http://gustavovalencia.net/app/webroot/img/Documents/BI/Actividades/001/Articulo DM.pdf

- Zagal Flores Roberto (2008). Alineación De Ontologías Usando El Método Boosting. *Instituto Politécnico Nacional*. Retrieved from http://levashkin.com/files/Roberto Eswart Zagal Flores.pdf

ANEXOS

ANEXO 1

Cargar de manera automática las ontologías al servidor virtuoso

```
<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<?php
/*define('__ROOT__', dirname(dirname(__FILE__)));
include(__ROOT__.'/ontoApi/core/examples/tablas.php');*/
?>
<html><!--lang="en"-->
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Analisis Lexico</title>

    <!-- Bootstrap Core CSS -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom CSS -->
    <link href="css/sb-admin.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <link href="font-awesome/css/font-awesome.min.css" rel="stylesheet"
type="text/css">
<script>
    function upload() {
        var xmlhttp = new XMLHttpRequest();
        var url = "core/examples/graficos.php";
        var data = new FormData(document.getElementById("form_ingreso"));
        xmlhttp.onreadystatechange=function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                 var array = JSON.parse(xmlhttp.responseText);
                if(array.resultado == 1){
                    //window.open("http://127.0.0.1/sc/index.php");
                    //document.location = "index.php";
                    alert("Archivo almacenado");
                }else{
                    alert("Error en la carga del archivo");
                }
            }
}
        xmlhttp.open("POST", url, true);
        xmlhttp.send(data);
    }
```

```php
<?php

$nombre = $_POST['nombre'];
$ubicacion = $_FILES["ubicacion"]["tmp_name"];

$ubicacion = str_replace("\\", "/", $ubicacion);


$first = "C:\Program Files\OpenLink Software\Virtuoso 7.2\bin\isql.exe";
$second = "isql -U dba -P dba -S 1111 EXEC=";
$third =
"DB.DBA.RDF_LOAD_RDFXML_MT(file_to_string_output('".$ubicacion."'),'','".
$nombre."');";
//echo '"'. $first .'" ' . $second.'"' .$third.'"' ;

$out = shell_exec('"'. $first .'" ' . $second.'"' .$third.'"' );

if (strpos($out, 'Done') !== false) {
    //echo 'true';
    $result["resultado"]= 1;
}else{
       //echo "false";
       $result["resultado"]= 0;
}

if( !$result ) { print sparql_errno() . ": " . sparql_error(). "\n";
exit; }

echo json_encode($result);

?>
```

ANEXO 2

Matriz de similitud léxica directa entre clases

Ontología 1	Ontología 2										
	Autobiography	**Biography**	**Essay**	**Human**	**LiteraryCritic**	**Literature**	**Novel**	**Poetry**	**Politics**	**Volume**	**Writer**
Book	11	7	5	5	14	10	4	5	7	5	6
CD	13	9	5	5	13	10	5	6	8	6	6
Children	11	8	8	7	13	9	7	7	8	6	6
DVD	13	9	5	5	14	10	5	6	8	6	6
Person	12	8	5	5	12	8	6	5	7	6	6
Pocket	12	8	6	6	13	9	4	4	6	5	5
Popular	11	8	6	5	12	8	5	5	6	5	6
Product	12	8	7	7	12	8	6	5	6	5	6
Publisher	11	9	8	8	13	9	8	8	6	7	6
Science	12	9	7	6	13	8	6	6	6	6	6
Textbook	11	9	8	8	13	9	8	7	8	8	7

ANEXO 3

Matriz de Afinidad entre clases

Ontología 1	Ontología 2										
	Autobiography	**Biography**	**Essay**	**Human**	**LiteraryCritic**	**Literature**	**Novel**	**Poetry**	**Politics**	**Volume**	**Writer**
Book	0.153	0.222	0	0	0	0	0.2	0.166	0.125	0.166	0
CD	0	0	0	0	0.071	0	0	0	0	0	0
Children	0.153	0.111	0	0.125	0.071	0.1	0.125	0.125	0	0.25	0.25
DVD	0	0	0	0	0	0	0	0	0	0	0
Person	0.076	0.111	0.166	0.166	0.142	0.2	0	0.166	0.125	0	0
Pocket	0.076	0.111	0	0	0.071	0.1	0.333	0.333	0.25	0.166	0.166
Popular	0.153	0.111	0.142	0.285	0.142	0.2	0.285	0.285	0.25	0.285	0.142
Product	0.076	0.111	0	0	0.142	0.2	0.142	0.285	0.25	0.285	0.142
Publisher	0.153	0	0.111	0.111	0.071	0.1	0.111	0.111	0.333	0.222	0.333
Science	0.076	0	0	0.142	0.071	0.2	0.142	0.142	0.25	0.142	0.142
Textbook	0.153	0	0	0	0.071	0.1	0	0.125	0	0	0.125

ANEXO 4

Matriz de similitud entre propiedades

Ontología 1	Ontología 2										
	Autobiography	**Biography**	**Essay**	**Human**	**LiteraryCritic**	**Literature**	**Novel**	**Poetry**	**Politics**	**Volume**	**Writer**
Book	0	0	0	0	0	0	0	0	0	0	0
CD	0	0	0	0	0	0	0	0	0	0	0
Children	0	0	0	0	0	0	0	0	0	0	0
DVD	0	0	0	0	0	0	0	0	0	0	0
Person	0	0	0	0	0	0	0	0	0	0	0
Pocket	0	0	0	0	0	0	0	0	0	0	0
Popular	0	0	0	0	0	0	0	0	0	0	0
Product	0	0	0	0.833	0	0	0	0	0	0	0
Publisher	0	0	0	0	0	0	0	0	0	0	0
Science	0	0	0	0	0	0	0	0	0	0	0
Textbook	0	0	0	0	0	0	0	0	0	0	0

ANEXO 5

Matriz de similitud entre superclases

Ontologia 1	Ontologia 2										
	Autobiography	**Biography**	**Essay**	**Human**	**LiteraryCritic**	**Literature**	**Novel**	**Poetry**	**Politics**	**Volume**	**Writer**
Book	0	0	0.142	0	0	0.142	0.1	0.1	0	0	0.416
CD	0	0	0.142	0	0	0.142	0.1	0.1	0	0	0.416
Children	0	0	0.083	0	0	0.083	0	0	0	0	0
DVD	0	0	0.142	0	0	0.142	0.1	0.1	0	0	0.416
Person	0	0	0	0	0	0	0	0	0	0	0
Pocket	0	0	0.083	0	0	0.083	0	0	0	0	0
Popular	0	0	0.071	0	0	0.071	0.1	0.1	0	0	0.071
Product	0	0	0	0	0	0	0	0	0	0	0
Publisher	0	0	0	0	0	0	0	0	0	0	0
Science	0	0	0.083	0	0	0.083	0	0	0	0	0
Textbook	0	0	0.071	0	0	0.071	0.1	0.1	0	0	0.071

ANEXO 6

Código de la aplicación Análisis comparativo para el enriquecimiento de estructuras semánticas

Index.php

```
1. <script type="text/javascript"
   src="https://www.gstatic.com/charts/loader.js"></script>
2.
3. <?php
4. /*define('__ROOT__', dirname(dirname(__FILE__)));
5. include(__ROOT__.'/ontoApi/core/examples/tablas.php');*/
6. ?>
7. <html><!--lang="en"-->
8.
9. <head>
10.
11.     <meta charset="utf-8">
12.     <meta http-equiv="X-UA-Compatible" content="IE=edge">
13.     <meta name="viewport" content="width=device-width, initial-scale=1">
14.     <meta name="description" content="">
15.     <meta name="author" content="">
16.
17.     <title>Analisis Lexico</title>
18.
19.     <!-- Bootstrap Core CSS -->
20.     <link href="css/bootstrap.min.css" rel="stylesheet">
21.
22.     <!-- Custom CSS -->
23.     <link href="css/sb-admin.css" rel="stylesheet">
24.
25.     <!-- Custom Fonts -->
26.     <link href="font-awesome/css/font-awesome.min.css" rel="stylesheet"
   type="text/css">
27.
28.     <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and
   media queries -->
29.     <!-- WARNING: Respond.js doesn't work if you view the page via
   file:// -->
30.     <!--[if lt IE 9]>
31.         <script
   src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
32.         <script
   src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></scrip
   t>
33.     <![endif]-->
34.     <script type="text/javascript" src="js/base64.js"></script>
35.     <script type="text/javascript" src="js/canvas2image.js"></script>
36.     <script type="text/javascript" src="js/html2canvas.js"></script>
37.
38.     <script>
39.
40.     (function() {
41.         var xmlhttp = new XMLHttpRequest();
```

```
42.  var url = "http://127.0.0.1/sc/core/examples/grafos.php";
43.
44.         //var data = new
    FormData(document.getElementById("form_ingreso"));
45.         xmlhttp.onreadystatechange=function() {
46.             if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
47.                  var array = JSON.parse(xmlhttp.responseText);
48.
49.                  var out = "<table cellpadding='0' cellspacing='0'
    border='1' class='display' id='example' width='100%'>";
50.                      for(var i = 0; i < array.grafos.length; i++) {
51.                          out += "<tr>";out += "<td>";
52.                          out += "<input type='radio' name='grafo1'
    id='grafo1' value='"+array.grafos[i]+"'>";out += "</td>";
53.                          out += "<td>"+array.grafos[i] +"</td>";
54.                          out += "</tr>";
55.                      }
56.                      out += "</tr>";
57.                      out += "</table>";
58.
59.                 document.getElementById("tabla_grafos1").innerHTML = out;
60.
61.                  var out = "<table cellpadding='0' cellspacing='0'
    border='1' class='display' id='example' width='100%'>";
62.                      for(var i = 0; i < array.grafos.length; i++) {
63.                          out += "<tr>";out += "<td>";
64.                          out += "<input type='radio' name='grafo2'
    id='grafo2' value='"+array.grafos[i]+"'>";out += "</td>";
65.                          out += "<td>"+array.grafos[i] +"</td>";
66.                          out += "</tr>";
67.                      }
68.                      out += "</tr>";
69.                      out += "</table>";
70.
71.                 document.getElementById("tabla_grafos2").innerHTML = out;
72.             }
73.         }
74.         xmlhttp.open("POST", url, true);
75.         xmlhttp.send();
76.     })();
77.
78.
79.     function openontology() {
80.         var xmlhttp = new XMLHttpRequest();
81. var url = "http://127.0.0.1/sc/core/examples/openontology.php";
82.         var data = new FormData(document.getElementById("form_ingreso"));
83.         xmlhttp.onreadystatechange=function() {
84.             if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
85.                  var array = JSON.parse(xmlhttp.responseText);
86.                  var out = "<table cellpadding='0' cellspacing='0'
    border='1' class='display' id='example' width='100%'>";
87.                      out += "<thead><tr><th rowspan='2'>Ontologia
    1</th><th colspan="+array.clase1.length+">Ontologia 2</th></tr></thead>";
88.                      out += "<tr>";
89.                      out += "<td></td>";
90.                      for(var i = 0; i < array.clase2.length; i++)
```

```
91. out += "<td>"+array.clase2[i] +"</td>";
92.                     }
93.                     out += "</tr>";
94.
95.                     var cont=0; var cont_clase1=0;
96.                     for(var i = 0; i < array.distancia_clase.length; i++)
    {
97.                         if(cont == 0){out += "<tr>";out +=
    "<td>"+array.clase1[cont_clase1] +"</td>";cont_clase1++;}
98.                         out += "<td>"+array.distancia_clase[i] +"</td>";
99.                         cont++;
100.                              if(cont == array.clase2.length){out +=
    "</tr>";cont=0;}
101.                          }
102.                          out += "</tr>";
103.                          out += "</table>";
104.
105.
    document.getElementById("tabla_distancia").innerHTML = out;
106.
107.                    var out = "<table cellpadding='0' cellspacing='0'
    border='1' class='display' id='example' width='100%'>";
108.                          out += "<thead><tr><th rowspan='2'>Ontologia
    1</th><th colspan="+array.clase1.length+">Ontologia 2</th></tr></thead>";
109.                          out += "<tr>";
110.                          out += "<td></td>";
111.                          for(var i = 0; i < array.clase2.length; i++) {
112.                               out += "<td>"+array.clase2[i] +"</td>";
113.                          }
114.                          out += "</tr>";
115.
116.                          var cont=0; var cont_clase1=0;
117.                          for(var i = 0; i <
    array.afinidad_clase.length; i++) {
118.                              if(cont == 0){out += "<tr>";out +=
    "<td>"+array.clase1[cont_clase1] +"</td>";cont_clase1++;}
119.                              out += "<td>"+array.afinidad_clase[i]
    +"</td>";
120.                              cont++;
121.                              if(cont == array.clase2.length){out +=
    "</tr>";cont=0;}
122.                          }
123.                          out += "</tr>";
124.                          out += "</table>";
125.
    document.getElementById("tabla_afinidad").innerHTML = out;
126.      var out = "<table cellpadding='0' cellspacing='0' border='1'
    class='display' id='example' width='100%'>";
127.                          out += "<thead><tr><th
    rowspan='2'>Ontología1</th><th colspan="+array.clase1.length+">Ontologia
    2</th></tr></thead>";
128.                          out += "<tr>";
129.                          out += "<td></td>";
130.                          for(var i = 0; i < array.clase2.length; i++) {
131.                               out += "<td>"+array.clase2[i] +"</td>";
132.                          } out += "</tr>"; }
```

```
133.      var cont=0; var cont_clase1=0;
134.                          for(var i = 0; i < array.afinidad_prop.length;
   i++) {
135.                              if(cont == 0){out += "<tr>";out +=
   "<td>"+array.clase1[cont_clase1] +"</td>";cont_clase1++;}
136.                              out += "<td>"+array.afinidad_prop[i]
   +"</td>";
137.                              cont++;
138.                              if(cont == array.clase2.length){out +=
   "</tr>";cont=0;}
139.                          }
140.                          out += "</tr>";
141.                          out += "</table>";
142.
   document.getElementById("similitud_propiedades").innerHTML = out;
143.      var out = "<table cellpadding='0' cellspacing='0' border='1'
   class='display' id='example' width='100%'>";
144.                          out += "<thead><tr><th rowspan='2'>Ontologia
   1</th><th colspan="+array.clase1.length+">Ontologia 2</th></tr></thead>";
145.                          out += "<tr>";
146.                          out += "<td></td>";
147.                          for(var i = 0; i < array.clase2.length; i++) {
148.                               out += "<td>"+array.clase2[i] +"</td>";
149.                          }
150.                          out += "</tr>";
151.
152.                          var cont=0; var cont_clase1=0;
153.                          for(var i = 0; i < array.superClass.length;
   i++) {
154.                              if(cont == 0){out += "<tr>";out +=
   "<td>"+array.clase1[cont_clase1] +"</td>";cont_clase1++;}
155.                              out += "<td>"+array.superClass[i]
   +"</td>";
156.                              cont++;
157.                              if(cont == array.clase2.length){out +=
   "</tr>";cont=0;}
158.                          }
159.                          out += "</tr>";
160.                          out += "</table>";
161.
   document.getElementById("similitud_superClases").innerHTML = out;
162.        //alineacion
163.                      var out = "<table cellpadding='0' cellspacing='0'
   border='1' class='display' id='example' width='100%'>";
164.                          out += "<thead><tr><th>TIPO DE
   SIMILITUD</th><th colspan='2' align='center'>PAREJAS</th></tr></thead>";
165.                          out += "<tr>";
166.                          out += "<td
   rowspan="+array.clase_F_Ont1.length+" align='center'
   width='25%'>SIMILITUD LÉXICA</td>";
167.                          for(var i = 0; i < array.clase_F_Ont1.length;
   i++) {
168.                              out += "<td align='center'
   width='25%'>"+array.clase_F_Ont1[i] +"</td>";
169.                              out += "<td align='center'
   width='25%'>"+array.clase_F_Ont2[i] +"</td>";
```

```
170.      out += "<tr>";
171.                          }
172.                          out += "<tr>";
173.                          out += "<td
   rowspan="+array.propi_F_Ont1.length+" align='center'>SIMILITUD ENTRE
   PROPIEDADES</td>";
174.                          for(var i = 0; i < array.propi_F_Ont1.length;
   i++) {
175.                              out += "<td
   align='center'>"+array.propi_F_Ont1[i] +"</td>";
176.                              out += "<td
   align='center'>"+array.propi_F_Ont2[i] +"</td>";
177.                              out += "<tr>";
178.                          }
179.                          out += "<tr>";
180.                          out += "<td
   rowspan="+array.super_F_Ont1.length+" align='center'>SIMILITUD ENTRE
   SUPERCLASES</td>";
181.                          for(var i = 0; i < array.super_F_Ont1.length;
   i++) {
182.                              out += "<td
   align='center'>"+array.super_F_Ont1[i] +"</td>";
183.                              out += "<td
   align='center'>"+array.super_F_Ont2[i] +"</td>";
184.                              out += "<tr>";
185.                          }out += "<tr>";
186.                          out += "</tr>";
187.                          out += "</table>";
188.                      document.getElementById("alineacion").innerHTML =
   out;
189.      //enlazado
190.                      var out = "<table cellpadding='0' cellspacing='0'
   border='1' class='display' id='example' width='100%'>";
191.                          out += "<thead><tr><th>TIPO DE
   SIMILITUD</th><th colspan='2' align='center'>PAREJAS</th></tr></thead>";
192.                          out += "<tr>";
193.                          out += "<td
   rowspan="+array.clase_en_Ont1.length+" align='center' width='35%'>Numero
   de enlaces "+array.numEnl+" con porcentaje minimo de similitud
   "+array.porcentEnl+"</td>";
194.                          for(var i = 0; i < array.clase_en_Ont1.length;
   i++) {
195.                              out += "<td align='center'
   width='25%'>"+array.clase_en_Ont1[i] +"</td>";
196.                              out += "<td align='center'
   width='25%'>"+array.clase_en_Ont2[i] +"</td>";
197.                              out += "<tr>";
198.                          }
199.                          out += "<tr>";
200.                          out += "</tr>";
201.                          out += "</table>";
202.
203.      document.getElementById("enlazado").innerHTML = out;
```

```
204.      /*Grafica ontologia1*/
205.                      google.charts.load('current',
   {packages:["orgchart"]});
206.                      google.charts.setOnLoadCallback(drawChartont1);
207.
208.                      function drawChartont1() {
209.                          var data = new
   google.visualization.DataTable();
210.                          data.addColumn('string', 'Name');
211.                          data.addColumn('string', 'Manager');
212.
213.                          // For each orgchart box, provide the name,
   manager, and tooltip to show.
214.                          var datos = new Array();
215.                          for (var i = 0; i < array.graf1A.length; i++)
   {
216.                              var cl1 = array.graf1B[i];
217.                              var cl2 = array.graf1A[i];
218.                              //cl2=cl2+" ";
219.
220.                              datos.push([cl1,cl2]);
221.                          }
222.                          //console.log(datos);
223.                          data.addRows(datos);
224.      // Create the chart.
225.                          var chart = new
   google.visualization.OrgChart(document.getElementById('chart_ont1'));
226.                          // Draw the chart, setting the allowHtml
   option to true for the tooltips.
227.                          chart.draw(data, {allowHtml:true, width:
   '100%', height: '100%'});
228.                      }
229.                      /*Fin grafica ontologia1*/
230.
231.                      /*Grafica ontologia2*/
232.                      google.charts.load('current',
   {packages:["orgchart"]});
233.                      google.charts.setOnLoadCallback(drawChartont2);
234.
235.                      function drawChartont2() {
236.                          var data = new
   google.visualization.DataTable();
237.                          data.addColumn('string', 'Name');
238.                          data.addColumn('string', 'Manager');
239.        // For each orgchart box, provide the name, manager, and tooltip
   to show.
240.                          var datos = new Array();
241.                          for (var i = 0; i < array.graf1A.length; i++)
   {
242.                              var cl1 = array.graf2B[i];
243.                              var cl2 = array.graf2A[i];
244.                              //cl2=cl2+" ";
245.
246.                              datos.push([cl1,cl2]);
247.                          }
```

```
248.      //console.log(datos);
249.                          data.addRows(datos);
250.
251.                          // Create the chart.
252.                          var chart = new
   google.visualization.OrgChart(document.getElementById('chart_ont2'));
253.                          // Draw the chart, setting the allowHtml
   option to true for the tooltips.
254.                          chart.draw(data, {allowHtml:true, width:
   '100%', height: '100%'});
255.                      }
256.                      /*Fin grafica ontologia2*/
257.        /*Grafica de enlazado*/
258.                      google.charts.load('current',
   {packages:["orgchart"]});
259.                      google.charts.setOnLoadCallback(drawChart);
260.
261.                      function drawChart() {
262.                          var data = new
   google.visualization.DataTable();
263.                          data.addColumn('string', 'Name');
264.                          data.addColumn('string', 'Manager');
265.
266.                          // For each orgchart box, provide the name,
   manager, and tooltip to show.
267.                          var datos = new Array();var es=" ";
268.                          for (var i = 0; i <
   array.clase_en_Ont1.length; i++) {
269.                              var cl1 =
   array.clase_en_Ont1[i];cl1=es+cl1;es += " ";
270.                              var cl2 = array.clase_en_Ont2[i];
271.                              cl2=cl2+" ";
272.
273.                              datos.push([cl1,cl2]);
274.                          }
275.                          //console.log(datos);
276.                          data.addRows(datos);
277.      // Create the chart.
278.                          var chart = new
   google.visualization.OrgChart(document.getElementById('chart_div'));
279.                          // Draw the chart, setting the allowHtml
   option to true for the tooltips.
280.                          chart.draw(data, {allowHtml:true, width:
   '100%', height: '100%'});
281.
282.                          html2canvas($("#widget"), {
283.                              onrendered: function(canvas) {
284.                                  theCanvas = canvas;
285.                                  document.body.appendChild(canvas);
286.        $("#img-out").append(canvas);
287.                                  // Clean up
288.                                  //document.body.removeChild(canvas);
289.                              }
290.                          });
291.                      } /*Fin grafica enlazado*/
```

```
292.      /*Grafica de mezclado*/
293.                      google.charts.load('current',
   {packages:["orgchart"]});
294.                      google.charts.setOnLoadCallback(drawChart1);
295.
296.                      function drawChart1() {
297.                          var data = new
   google.visualization.DataTable();
298.                          data.addColumn('string', 'Name');
299.                          data.addColumn('string', 'Manager');
300.
301.
302.                          // For each orgchart box, provide the name,
   manager, and tooltip to show.
303.                          var datos = new Array();
304.                          for (var i = 0; i < array.padre.length; i++) {
305.                              var cl1 = array.subClass[i];cl1=cl1+" ";
306.                              var cl2 = array.padre[i];cl2=cl2+" ";
307.                              datos.push([cl1,cl2]);
308.                          }
309.
310.                          data.addRows(datos);
311.        // Create the chart.
312.                          var chart1 = new
   google.visualization.OrgChart(document.getElementById('chart_div_mix'));
313.                          // Draw the chart, setting the allowHtml
   option to true for the tooltips.
314.                          chart1.draw(data, {allowHtml:true, width:
   '100%', height: '100%'});
315.
316.                          html2canvas($("#widget1"), {
317.                              onrendered: function(canvas) {
318.                                  theCanvas = canvas;
319.                                  document.body.appendChild(canvas);
320.
321.                                  //Canvas2Image.saveAsPNG(canvas);
322.                                  $("#img-out1").append(canvas);
323.                                  // Clean up
324.                                  //document.body.removeChild(canvas);
325.                              }
326.                          });
327.                      }
328.                      /*Fin grafica mezclado*/
329.                  }
330.              }
331.          xmlhttp.open("POST", url, true);
332.          xmlhttp.send(data);
333.          }
334.      function ACT1() {
335.              var div1 = document.getElementById('tabla_distancia');
336.              if(div1.style.display == 'block'){
337.                  div1.style.display = 'none';
338.              }else{
339.                  div1.style.display = 'block';
340.              }
341.          }
```

```
342.      function ACT2() {
343.              var div1 = document.getElementById('tabla_afinidad');
344.              if(div1.style.display == 'block'){
345.                  div1.style.display = 'none';
346.              }else{
347.                  div1.style.display = 'block';
348.              }
349.          }
350.          function ACT3() {
351.              var div1 =
    document.getElementById('similitud_propiedades');
352.              if(div1.style.display == 'block'){
353.                  div1.style.display = 'none';
354.              }else{
355.                  div1.style.display = 'block';
356.              }
357.          }
358.          function ACT4() {
359.              var div1 =
    document.getElementById('similitud_superClases');
360.              if(div1.style.display == 'block'){
361.                  div1.style.display = 'none';
362.              }else{
363.                  div1.style.display = 'block';
364.              }
365.          }
366.          function ACT5() {
367.              var div1 = document.getElementById('alineacion');
368.              if(div1.style.display == 'block'){
369.                  div1.style.display = 'none';
370.              }else{
371.                  div1.style.display = 'block';
372.              }
373.          }
374.      function ACT6() {
375.              var div1 = document.getElementById('enlazado');
376.              if(div1.style.display == 'block'){
377.                  div1.style.display = 'none';
378.              }else{
379.                  div1.style.display = 'block';
380.              }
381.          }
382.          </script>
383.      </head>
384.
385.      <body>
386.
387.          <div id="wrapper">
388.      <nav class="navbar navbar-inverse navbar-fixed-top"
    role="navigation">
389.                  <!-- Brand and toggle get grouped for better mobile
    display -->
390.                  <div class="navbar-header">
391.                      <button type="button" class="navbar-toggle" data-
    toggle="collapse" data-target=".navbar-ex1-collapse">
392.                          <span class="sr-only">Toggle navigation</span>
```

```
393.      <span class="icon-bar"></span>
394.                          <span class="icon-bar"></span>
395.                          <span class="icon-bar"></span>
396.                      </button>
397.                      <a class="navbar-brand"
   href="index.php">ANALIZADOR DE ONTOLOGÍAS</a>
398.                      <span>|</span>
399.                      <a class="navbar-brand"
   href="uploadowl.php">CARGAR ONTOLOGÍAS</a>
400.                  </div>
401.      </div>
402.                  <!-- /.navbar-collapse -->
403.              </nav>
404.              <div id="page-wrapper">
405.                  <div class="container-fluid">
406.
407.                      <!-- Page Heading -->
408.                      <div class="row">
409.                          <div class="col-lg-12">
410.                              <h1 class="page-header">
411.                                <div align="center"><b>Técnicas de
   Minería Ontológica</b> <small> (Sparqllib)</small>
412.                              </h1>
413.                          </div>
414.                      </div>
415.                      <!-- /.row -->
416.
417.                      <!-- informacion -->
418.                      <div class="row">
419.                          <form id="form_ingreso" name="form_ingreso"
   href="javascript:;" action="openontology();return false;">
420.                              <div class="col-lg-6">
421.                                  <div class="panel panel-primary
   class">
422.                                      <div class="panel-heading">
423.                                          <h3 class="panel-
   title">Seleccione ontología - <small> Grafo 1</small></h3>
424.                                      </div>
425.                                      <div class="panel-body">
426.                                          <!--<label><input
   name="grafo1" type="text" id="grafo1" width="100%"/></label> -->
427.                                              <div class="table-
   responsive">
428.                                                  <div
   id='tabla_grafos1'></div>
429.                                              </div>
430.                                        </div>
431.      </div>
432.                              </div>
433.                              <div class="col-lg-6">
434.                                  <div class="panel panel-primary
   class">
435.       <div class="panel-heading">
436.      <h3 class="panel-title">Seleccione ontología - <small> Grafo
   2</small></h3>
437.                                      </div>
```

```
438.      <div class="panel-body">
439.                                          <!--<label><input
   name="grafo2" type="text" id="grafo2" /></label> -->
440.                                              <div class="table-
   responsive">
441.                                                  <div
   id='tabla_grafos2'></div>
442.                                              </div>
443.                                      </div>
444.                                  </div>
445.                              </div>
446.                              <div class="col-lg-6">
447.                                  <label><input type="button"
   href="javascript:;" onclick="openontology();return false;" class="btn
   btn-success" value="ANÁLISIS ONTOLÓGICO"/></label>
448.                              </div>
449.
450.                          </form>
451.                      </div>
452.                      <!-- /.row -->
453.
454.                      <div class="row">
455.
456.                          <div class="col-lg-6">
457.                              <h4><b>Grafo Original Ontología 1</b></h4>
458.                              <div class="table-responsive">
459.                                  <div id="chart_ont1"></div>
460.                              </div>
461.                          </div>
462.                          <div class="col-lg-6">
463.                              <h4><b>Grafo Original ontología 2</b></h4>
464.                              <div class="table-responsive">
465.                                  <div id="chart_ont2"></div>
466.                              </div>
467.                          </div>
468.                      </div>
469.      <div class="row">
470.                          <div class="col-lg-12">
471.                              <h2><div align="center"><b>Medidas De
   Similitud De Ontologías</b></h2>
472.                              <h2>Similitud Directa entre Clases:</h2>
473.                              <div onclick="ACT1()"><a id="accionar">
   Ver | Ocultar</a> </div>
474.                              <div class="table-responsive">
475.                                  <div id='tabla_distancia'
   style="display:none;"></div>
476.                              </div>
477.                          </div>
478.      <div class="col-lg-12">
479.                              <h2>Similitud de Afinidad entre
   Clases:</h2>
480.                              <div onclick="ACT2()"><a id="accionar">
   Ver | Ocultar</a> </div>
481.      <div class="table-responsive">
482.                                  <div id='tabla_afinidad'
   style="display:none;"></div>
```

```
483.      </div>
484.                          </div>
485.
486.                          <div class="col-lg-12">
487.                              <h2>Similitud entre Propiedades:</h2>
488.                              <div onclick="ACT3()"><a id="accionar">
   Ver | Ocultar</a> </div>
489.                              <div class="table-responsive">
490.                                  <div id='similitud_propiedades'
   style="display:none;"></div>
491.                              </div>
492.                          </div>
493.        <div class="col-lg-12">
494.                              <h2>Similitud entre SuperClases:</h2>
495.                              <div onclick="ACT4()"><a id="accionar">
   Ver | Ocultar</a> </div>
496.                              <div class="table-responsive">
497.                                  <div id='similitud_superClases'
   style="display:none;"></div>
498.                              </div>
499.                          </div>
500.
501.                          <div class="col-lg-12">
502.                              <h2><div align="center"><b>Alineación De
   Ontologías</b></h2>
503.                              <div onclick="ACT5()"><a id="accionar">
   Ver | Ocultar</a> </div>
504.                              <div class="table-responsive">
505.                                  <div id='alineacion'
   style="display:none;"></div>
506.                              </div>
507.                          </div>
508.
509.                          <div class="col-lg-12">
510.                              <h2><div align="center"><b>Enlazado De
   Ontologías</b></h2>
511.                              <div onclick="ACT6()"><a id="accionar">
   Ver | Ocultar</a> </div>
512.                              <div class="table-responsive">
513.                                  <div id='enlazado'
   style="display:none;"></div>
514.                              </div>
515.                              <div class="table-responsive">
516.                                  <h2>Resultado Ontología Enlazada</h2>
517.                                  <span id="widget" class="widget"
   field="owl" roundby="10" description="Enlazado">
518.                                      <div id="chart_div"></div>
519.                                  </span>
520.                                  <h3>Descargar Grafo Ontología
   Enlazada</h3>
521.                                  <div id="img-out"></div>
522.                              </div>
523.      </div>
524.
525.                          <div class="col-lg-12">
```

```
526.      <h2><div align="center"><b>Mezclado De Ontologías</b></h2>
527.                               <h2>Resultado Ontología Mezclada</h2>
528.                               <div class="table-responsive">
529.                                   <span id="widget1" class="widget1"
   field="owl" roundby="20" description="Mezclado">
530.                                       <div id="chart_div_mix"></div>
531.                                   </span>
532.                                   <h3>Descargar Grafo Ontología
   Mezclada</h3>
533.                                   <div id="img-out1"></div>
534.                               </div>
535.                           </div>
536.
537.                       </div>
538.      </div>
539.                  <!-- /.container-fluid -->
540.
541.              </div>
542.              <!-- /#page-wrapper -->
543.
544.          </div>
545.          <!-- /#wrapper -->
546.
547.          <!-- jQuery -->
548.          <script src="js/jquery.js"></script>
549.
550.          <!-- Bootstrap Core JavaScript -->
551.          <script src="js/bootstrap.min.js"></script>
552.
553.      </body>
554.
555.      </html>
```

Knn_adabost.php

```
<?php
function AdaBoost_Knn ($Matriz_similitud, $num_clases1, $num_clases2){
$t=1;
$T=10;
$umbral_pertenencia=0.1;//0.25   0.1  0.08
$desviacion_ini=1;
$rango_solucion=array(0,1);
$iteraciones=10;
$umbral=15;


      $D = K_Vecinos($Matriz_similitud,
$umbral_pertenencia,$desviacion_ini, $rango_solucion,
$iteraciones,$umbral, $num_clases1, $num_clases2);

      if($D != null){
            $datos2 = array();
            //if($D != 0){
            $F = $D[0];
            $d = $D[1];
            $distEucla = $D[2];

            $cont=0;$i=0;

            for ($i=0; $i < sizeof($Matriz_similitud); $i++) {
                  foreach ($d as &$debiles) {
                        if($i==$debiles){
                              $datos2[$cont]=$Matriz_similitud[$i];
                              $cont++;
                              break;
                        }
                  }
            }
            $desviacion_ini = 1*0.5*$iteraciones;

            $De = K_Vecinos($Matriz_similitud,
$umbral_pertenencia,$desviacion_ini, $rango_solucion,
$iteraciones,$umbral, $num_clases1, $num_clases2);
            $D = $De[0];

            return $Analisis = [$F,$D,$distEucla];
}else{
            return 0;
      }

}


$dist_euclidiana = array();
```

```
46. function K_Vecinos($Matriz_similitud,
    $umbral_pertenencia,$desviacion_ini, $rango_solucion,
    $iteraciones,$umbral, $num_clases1, $num_clases2){
47.       //$Matriz_similitud=[1,2,3,1];
48.       $C =
    array(array(max($Matriz_similitud),$desviacion_ini));//$Matriz_similitud[
    0]
49.       $p = array();
50.       for ($h=0; $h < $iteraciones; $h++) {
51.             for ($i=0; $i < sizeof($Matriz_similitud); $i++) {
52.                   for ($j=0; $j < sizeof($C); $j++) {
53.                         $resta =$Matriz_similitud[$i]-$C[$j][0];
54.                         $potencia = pow($resta, 2);
55.                         $dist = $potencia/$C[$j][1];
56.                         if($dist < $umbral_pertenencia) {
57.                   $p[$i]=1;
58.             }else{
59.                               $p[$i]=0;
60.             }
61.                   }
62.             }
63.       }
64.
65.       $claseC = array();
66.       for ($i=0; $i < sizeof($Matriz_similitud); $i++) {
67.             for ($j=0; $j < sizeof($C); $j++) {
68.                   if(1 == $p[$i]){
69.                         array_push($claseC, $Matriz_similitud[$i]);
70.                   }
71.             }
72.       }
73.       $media = array_sum($claseC)/count($claseC);
74.       $desviacion = stats_standard_deviation($claseC);
75.       $C[0][0]=$media;
76.       $C[0][1]=$desviacion;
77.       //var_dump($C);
78.       $distEuclaFuerte = array();
79.             for ($h=0; $h < $iteraciones; $h++) {
80.                   for ($i=0; $i < sizeof($Matriz_similitud); $i++) {
81.                         for ($j=0; $j < sizeof($C); $j++) {
82.                               $resta =$Matriz_similitud[$i]-$C[$j][0];
83.                               $potencia = pow($resta, 2);
84.                               if($C[$j][1] != 0){
85.                                     $dist = $potencia/$C[$j][1];
86.                                     if($dist < $umbral_pertenencia) {
87.                               $p[$i]=1;
88.                               $distEuclaFuerte[$i]=  $dist;
89.                         }else{
90.                                                 $p[$i]=0;
91.                         }
92.                        }else{
93.                         $p[$i]=0;
94.                        }
95.                         }
96.                   }
97.             }
```

```
98. $marcar = array();
99.              $class = array();
100.                   for ($i=0; $i < sizeof($C); $i++) {
101.                          for ($j=0; $j < sizeof($p); $j++) {
102.                                 if(1 == $p[$j]){
103.                                        array_push($class, $p[$i]);
104.                                        $marcar[$i] = array_sum($class);
105.                                 }
106.                          }
107.                   }
108.
109.                   $lim = 0;
110.                   $C = array();$n = array();
111.                   for ($i=0; $i < sizeof($marcar); $i++) {
112.                          for ($j=0; $j < sizeof($p); $j++) {
113.                                 if($p[$j]>$lim){
114.                                        $C[$j]=$j;
115.                                 }else{
116.                                        $n[$j]=$j;
117.                                 }
118.                          }
119.                   }
120.
121.                   if(sizeof($n)>$umbral){
122.                          $C = [$C,$n,$distEuclaFuerte];
123.                   }
124.
125.                   return $C;
126.
127.      }
128.
129.
130.          function stats_standard_deviation(array $a, $sample = false) {
131.            $n = count($a);
132.            if ($n === 0) {
133.              trigger_error("The array has zero elements",
    E_USER_WARNING);
134.              return false;
135.            }
136.            if ($sample && $n === 1) {
137.              trigger_error("The array has only 1 element",
    E_USER_WARNING);
138.              return false;
139.            }
140.            $mean = array_sum($a) / $n;
141.            $carry = 0.0;
142.            foreach ($a as $val) {
143.              $d = ((double) $val) - $mean;
144.              $carry += $d * $d;
145.            };
146.            if ($sample) {
147.             --$n;
148.           }
149.           return sqrt($carry / $n);
150.         }
```

```
151.      function calcularAlineacion($al_clase,$afinClass,$clase2){
152.                  $c_f = array();
153.                  $cont=0;$cont1=0;$cont2=0;
154.                  $aux=0;
155.                  foreach ($al_clase[0] as &$value) {
156.                        for ($i=0; $i < sizeof($afinClass); $i++) {
157.                                    if($value == $cont){
158.                                          $c_f[$aux]=[$cont1,$cont2];
159.                                          $aux++;
160.                                          break;
161.                                    }
162.                                    if($cont2 == sizeof($clase2)-1){
163.                                          $cont1++;
164.                                          $cont2=0;
165.                                    }else{
166.                                          $cont2++;
167.                                    }
168.                                    $cont++;
169.                        }
170.                  }
171.                  return $c_f;
172.            }
173.
174.            function AliClasesOnt ($classAlin,$clase1,$clase2){
175.            $al_clases1 = array();$al_clases2 = array();
176.            for ($j=0; $j < sizeof($classAlin); $j++) {
177.                  for ($i=0; $i < sizeof($clase1); $i++) {
178.                        if($classAlin[$j][0] == $i){
179.                              array_push($al_clases1, $clase1[$i]);
180.                              break;
181.                        }
182.                  }
183.                  for ($k=0; $k < sizeof($clase2); $k++) {
184.                        if($classAlin[$j][1] == $k){
185.                              array_push($al_clases2, $clase2[$k]);
186.                              break;
187.                        }
188.                  }
189.            }
190.            return $ali = [$al_clases1,$al_clases2];
191.      }
192.
193.      ?>
```

Openontology.php

```
1. <?php
2. require_once('knn_adabost.php');
3. ?>
4. <?php
5. define('__ROOT__', dirname(dirname(__FILE__)));
6. require_once(__ROOT__.'/sparqllib.php');;
7. $db = sparql_connect( "http://localhost:8890/sparql" );   //conexion al
   endpoint d virtuoso mediante libreria de virtuoso
8. if( !$db ) { print sparql_errno() . ": " . sparql_error(). "\n"; exit; }
9. $graph1 = $_POST['grafo1'];
10. $graph2 = $_POST['grafo2'];
11. $clase1 = array();
12. $clase2 = array();
13. $distancia_clase = array();
14. $distClass = array();
15. $afinClass = array();
16. $afinProp = array();
17. $afinObj = array();
18. $simSuperClass = array();
19. //Clases
20. $sparql = "SELECT ?s FROM <".$graph1."> WHERE { ?s a owl:Class }";
21. $result1 = sparql_query( $sparql );
22. $max_class1 = sizeof($result1->rows);
23. for($i = 0; $i < $max_class1;$i++){
24.       $posicion_coincidencia = strrpos($result1->rows[$i]["s"]["value"],
   "nodeID://");
25.       $posicion_Thing = strrpos($result1->rows[$i]["s"]["value"],
   "Thing");
26.       if ($posicion_coincidencia === false) {
27.             if ($posicion_Thing === false) {
28.                   $class = generate_atributos($result1-
   >rows[$i]["s"]["value"]);
29.                   array_push($clase1, $class);
30.             }
31.       }
32. }
33. $result["clase1"]= $clase1;
34. $sparql = "SELECT ?s FROM <".$graph2."> WHERE { ?s a owl:Class }";
35. $result1 = sparql_query( $sparql );
36. $max_class2 = sizeof($result1->rows);
37. for($i = 0; $i < $max_class2;$i++){
38.       $posicion_coincidencia = strrpos($result1->rows[$i]["s"]["value"],
   "nodeID://");
39.       $posicion_Thing = strrpos($result1->rows[$i]["s"]["value"],
   "Thing");
40.       if ($posicion_coincidencia === false) {
41.             if ($posicion_Thing === false) {
42.                   $class = generate_atributos($result1-
   >rows[$i]["s"]["value"]);
43.                   array_push($clase2, $class);
44.             }
45.       }     }
46. $result["clase2"]= $clase2;
```

```
1. //Afinidad
2. for( $i=0;$i< sizeof($clase1);$i++){
3.     for( $j=0;$j< sizeof($clase2);$j++){
4.        $distancia=calcularDistancia($clase1[$i], $clase2[$j]);
5.        $afinidad=calcularAfinidad($clase1[$i], $clase2[$j], $distancia);
6.        $afinidad_pro = similitudPropiedades($clase1[$i], $clase2[$j],
   $graph1,$graph2);
7.        $superClass = similitudSuperClass($clase1[$i], $clase2[$j],
   $graph1, $graph2);
8.
9.        //$afinidad_obj = similitudObjectProperty($clase1[$i],
   $clase2[$j],$graph1,$graph2);
10.       array_push($distClass, $distancia);
11.       array_push($afinClass, $afinidad);
12.       array_push($afinProp, $afinidad_pro);
13.       array_push($simSuperClass, $superClass);
14.       //array_push($afinObj, $afinidad_obj);
15.    }
16.}
17.$result["distancia_clase"] = $distClass;
18.$result["afinidad_clase"] = $afinClass;
19.$result["afinidad_prop"] = $afinProp;
20.$result["superClass"] = $simSuperClass;
21.//$result["afinidad_obj"] = $afinObj;
22./*ADABOST Y Knn*/
23.$num_clases1 = sizeof($clase1);
24.$num_clases2  = sizeof($clase2);
25.$al_clase = AdaBoost_Knn($afinClass, $num_clases1, $num_clases2);
26.$al_propi = AdaBoost_Knn($afinProp, $num_clases1, $num_clases2);
27.$al_supcl = AdaBoost_Knn($simSuperClass, $num_clases1, $num_clases2);
28.
29./*Asignacion de posicion de fuertes*/
30.$classAlin = calcularAlineacion($al_clase,$afinClass,$clase2);
31.if($al_propi != 0){
32.       $propiAlin = calcularAlineacion($al_propi,$afinProp,$clase2);
33.}
34.if($al_supcl != 0){
35.$supclAlin = calcularAlineacion($al_supcl,$simSuperClass,$clase2);
36.}
37./*Alineacion Clases*/
38.$aclass1 = AliClasesOnt($classAlin,$clase1,$clase2);
39.if($al_propi != 0){
40.       $pclass1 = AliClasesOnt($propiAlin,$clase1,$clase2);
41.       $result["propi_F_Ont1"] = $pclass1[0];
42.       $result["propi_F_Ont2"] = $pclass1[1];
43.}else{
44.       $result["propi_F_Ont1"] = 0;
45.       $result["propi_F_Ont2"] = 0;
46.}
47.if($al_supcl != 0){
48.       $sclass1 = AliClasesOnt($supclAlin,$clase1,$clase2);
49.       $result["super_F_Ont1"] = $sclass1[0];
50.       $result["super_F_Ont2"] = $sclass1[1];
51.}else{
52.       $result["super_F_Ont1"] = 0;
53.       $result["super_F_Ont2"] = 0;
```

```
/*ENLAZADO*/
/*Porcentaje de similitud minimo*/
$pormin = 80;
$valMax = max($al_clase[2]);
$valorMin = ($pormin*$valMax)/100;
$i=0;$j=0;
foreach ($al_clase[2] as $value) {
	if($value>$valorMin){
		$en_clases1[$j] = $aclass1[0][$i];
		$en_clases2[$j] = $aclass1[1][$i];
		$j++;
	}
	$i++;
}
$result["porcentEnl"] = $pormin;
$result["numEnl"] = sizeof($en_clases1);
$result["clase_en_Ont1"] = $en_clases1;
$result["clase_en_Ont2"] = $en_clases2;
/*Algoritmo mezcla fuerte*/

/*Algoritmo mezcla debil*/
$claseC = $clase1;
//alineacion
$aliOntclass1 = $aclass1[0];
$aliOntclass2 = $aclass1[1];
$aliOntvalor2 = $al_clase[2];
	$padre = array();
	$subClass = array();$contA = 0;
for ($i=0; $i < sizeof($aliOntclass2); $i++) {
	$pref1 = "PREFIX a: <".$graph2.">";
$sparql = $pref1." SELECT * FROM <".$graph2."> WHERE { ?s rdfs:subClassOf
a:".$aliOntclass2[$i]." . }";
	$result1 = sparql_query( $sparql );
	$max_pro1 = sizeof($result1->rows);
	$cont = 0;
	if($max_pro1 > 0){
		while ($cont < $max_pro1) {
			$subclase = generate_atributos($result1-
>rows[$cont]["s"]["value"]);
			if (!in_array($subclase, $aliOntclass2)) {
				$padre[$contA] = "Thing";
				$subClass[$contA] = $subclase;
			}else{
				$padre[$contA] = "Thing";
				$subClass[$contA] = $aliOntclass2[$i];
				$contA++;
				$padre[$contA] = $aliOntclass2[$i];
				$subClass[$contA] = $subclase;
				}
				$cont++;
				$contA++;
			}
		}
	}

	/*FIN Algoritmo mezcla debil*/
```

```
108.      $CnoCA = array();$contA=0;
109.      for ($i=0; $i < sizeof($clase2); $i++) {
110.            $bandera = 1;
111.            for ($j=0; $j < sizeof($subClass); $j++) {
112.                  if($clase2[$i] == $subClass[$j]){
113.                        $bandera = 0;
114.                        break;
115.                  }
116.            }
117.            for ($j=0; $j < sizeof($padre); $j++) {
118.                  if($clase2[$i] == $padre[$j]){
119.                        $bandera = 0;
120.                        break;
121.                  }
122.            }
123.            if($bandera == 1){
124.                  $CnoCA[$contA] = $clase2[$i];
125.                  $contA++;
126.            }
127.      }
128.      if(sizeof($CnoCA) > 1){
129.            $contA=sizeof($subClass);
130.            for ($i=0; $i < sizeof($CnoCA); $i++) {
131.                  $pref1 = "PREFIX a: <".$graph2.">";
132.                  $sparql = $pref1." SELECT * FROM <".$graph2."> WHERE
   { a:".$CnoCA[$i]." rdfs:subClassOf ?s }";
133.                  $result1 = sparql_query( $sparql );
134.                  $max_pro1 = sizeof($result1->rows);
135.                  $cont = 0;
136.                  if($max_pro1 > 0){
137.                        while ($cont < $max_pro1) {
138.                              if($result1->rows[$cont]["s"]["type"] ==
   "uri"){
139.                                    $superClass =
   generate_atributos($result1->rows[$cont]["s"]["value"]);
140.                                    if(!is_null($superClass)){
141.                                          $padre[$contA] = "Thing";
142.                                          $subClass[$contA] =
   $superClass;
143.                                          $contA++;
144.                                          $padre[$contA] =
   $superClass;
145.                                          $subClass[$contA] =
   $CnoCA[$i];
146.                                          $contA++;
147.
148.
149.                                          $pref1 = "PREFIX a:
   <".$graph2.">";
150.
```

```
151.      $sparql = $pref1." SELECT * FROM <".$graph2."> WHERE { ?s
   rdfs:subClassOf a:".$CnoCA[$i]." . }";
152.                                                $result1 = sparql_query(
   $sparql );
153.                                                $max_pro1 =
   sizeof($result1->rows);
154.                                                $cont = 0;
155.                                                if($max_pro1 > 0){
156.                                                      while ($cont <
   $max_pro1) {
157.                                                            $subclase =
   generate_atributos($result1->rows[$cont]["s"]["value"]);
158.      if (in_array($subclase, $aliOntclass2)) {
                                    $padre[$contA] = $aliOntclass2[$i];
                                                      $subClass[$contA] =
   $subclase;
159.                                                      }
160.                                                      $cont++;
161.                                                      $contA++;
162.                                                }
163.                                          }
164.                                    }
165.                              }
166.                              $cont++;
167.                        }
168.                  }
169.            }
170.      }
171.      $contA=sizeof($subClass);
172.      for ($i=0; $i < sizeof($claseC); $i++) {
173.            $pref1 = "PREFIX a: <".$graph1.">";
174.            $sparql = $pref1." SELECT * FROM <".$graph1."> WHERE { ?s
   rdfs:subClassOf a:".$claseC[$i]." . }";
175.            $result1 = sparql_query( $sparql );
176.            $max_pro1 = sizeof($result1->rows);
177.            $cont = 0;
178.            if($max_pro1 > 0){
179.                  while ($cont < $max_pro1) {
180.                        $subclase = generate_atributos($result1-
   >rows[$cont]["s"]["value"]);
181.                        if (!in_array($subclase, $aliOntclass1)) {
182.                              $padre[$contA] = "Thing";
183.                              $subClass[$contA] = $subclase;
184.                        }else{
185.                              $padre[$contA] = $aliOntclass2[$i];
186.                              $subClass[$contA] = $subclase;
187.                        }
188.                        $cont++;
189.                        $contA++;
190.                  }
191.            }
192.      }
193.      //array_multisort($padre, $subClass);
194.      $result["padre"] = $padre;
195.      $result["subClass"] = $subClass;
196.      /*FIN Algoritmo mezcla fuerte*/
```

```
197.      /*Calcular afinidad*/
198.        function calcularAfinidad($claseOnt1, $claseOnt2, $distancia){
199.              $longitud=0;
200.              if(strlen($claseOnt1)>strlen($claseOnt2)){
201.                  $longitud=strlen($claseOnt1);
202.              }else{
203.                  $longitud=strlen($claseOnt2);
204.              }
205.              $afinidad = 1-($distancia/$longitud);
206.              return $afinidad;
207.          }
208.      /* FIN Calcular afinidad*/
209.
210.      /*Calcular distancia*/
211.      function calcularDistancia($claseOnt1,$claseOnt2){
212.          return
   computeLevenshteinDistance(str_split($claseOnt1),str_split($claseOnt2));
213.      }
214.
215.      function computeLevenshteinDistance($clase1, $clase2) {
216.             for( $i=0;$i<=sizeof($clase1);$i++){
217.                    $distancia_clase[$i][0]=$i;
218.             }
219.             for( $j=0;$j<=sizeof($clase2);$j++){
220.                    $distancia_clase[0][$j]=$j;
221.             }
222.                    for( $i=1;$i<=sizeof($clase1);$i++){
223.                   for( $j=1;$j<=sizeof($clase2);$j++){
224.                    if($clase1[$i-1]==$clase2[$j-
   1]){$aux=0;}else{$aux=1;}
225.                          $distancia_clase[$i][$j]=
   minimum($distancia_clase[$i-1][$j]+1,$distancia_clase[$i][$j-
   1]+1,$distancia_clase[$i-1][$j-1]+($aux));
226.                   }
227.              }
228.              return $val =
   $distancia_clase[sizeof($clase1)][sizeof($clase2)];
229.      }
230.      function minimum($a, $b, $c) {
231.              if($a<=$b && $a<=$c){
232.                  return $a;
233.              }
234.              if($b<=$a && $b<=$c){
235.                  return $b;
236.              }
237.              return $c;
238.      }
239.      /*FIN Calcular distancia*/
240.
```

```
241.      /*Propiedades (DatatypeProperty)*/
242.      function similitudPropiedades($claseOnt1, $claseOnt2, $graph1,
   $graph2) {
243.            $propiedad1 = array();
244.            $propiedad2 = array();
245.
246.            $pref1 = "PREFIX a: <".$graph1.">";
247.            $pref2 = "PREFIX a: <".$graph2.">";
248.
249.            $sparql = $pref1." SELECT * FROM <".$graph1."> WHERE { ?s a
   owl:DatatypeProperty . ?s rdfs:domain a:".$claseOnt1." }";
250.            $result1 = sparql_query( $sparql );
251.            $max_pro1 = sizeof($result1->rows);
252.            for($i = 0; $i < $max_pro1;$i++){
253.                  $pro = generate_atributos($result1-
   >rows[$i]["s"]["value"]);
254.                  array_push($propiedad1, $pro);
255.            }
256.
257.            $sparql = $pref2." SELECT * FROM <".$graph2."> WHERE { ?s a
   owl:DatatypeProperty . ?s rdfs:domain a:".$claseOnt2." }";
258.            $result2 = sparql_query( $sparql );
259.            $max_pro1 = sizeof($result2->rows);
260.            for($i = 0; $i < $max_pro1;$i++){
261.                  $pro = generate_atributos($result2-
   >rows[$i]["s"]["value"]);
262.                  array_push($propiedad2, $pro);
263.            }
264.            /*echo "Propiedades 1";
265.            print_r($propiedad1);
266.            echo "Propiedades 2";
267.            print_r($propiedad2);*/
268.
269.            $sum = 0; $sumT= 0; $sumAux = 0;
270.            for( $i=0;$i< sizeof($propiedad1);$i++){
271.            for( $j=0;$j< sizeof($propiedad2);$j++){
272.                  $distancia=calcularDistancia($propiedad1[$i],
   $propiedad2[$j]);
273.                  $afinidad=calcularAfinidad($propiedad1[$i],
   $propiedad2[$j], $distancia);
274.
275.                        if($afinidad == 1){
276.                        $sum = $sum + $afinidad;
277.                  }else{
278.                        $sumAux = $sumAux + $afinidad;
279.                  }
280.            }
281.            }
282.            if($sum > 0){
283.                  $sumT = $sum/($sum+$sumAux);
284.            }
285.            return $sumT;
```

```
286.      }/*Propiedades (ObjectProperty)*/
287.      function similitudObjectProperty($claseOnt1, $claseOnt2, $graph1,
   $graph2) {
288.            $propiedad1 = array();
289.            $propiedad2 = array();
290.
291.            $pref1 = "PREFIX a: <".$graph1.">";
292.            $pref2 = "PREFIX a: <".$graph2.">";
293.
294.            $sparql = $pref1." SELECT * FROM <".$graph1."> WHERE { ?s a
   owl:ObjectProperty . ?s rdfs:domain a:".$claseOnt1." }";
295.            $result1 = sparql_query( $sparql );
296.            $max_pro1 = sizeof($result1->rows);
297.            for($i = 0; $i < $max_pro1;$i++){
298.                  $pro = generate_atributos($result1-
   >rows[$i]["s"]["value"]);
299.                  array_push($propiedad1, $pro);
300.            }
301.
302.            $sparql = $pref2." SELECT * FROM <".$graph2."> WHERE { ?s a
   owl:ObjectProperty . ?s rdfs:domain a:".$claseOnt2." }";
303.            $result2 = sparql_query( $sparql );
304.            $max_pro1 = sizeof($result2->rows);
305.            for($i = 0; $i < $max_pro1;$i++){
306.                  $pro = generate_atributos($result2-
   >rows[$i]["s"]["value"]);
307.                  array_push($propiedad2, $pro);
308.            }
309.
310.            $sum = 0; $sumT= 0; $sumAux = 0;
311.            for( $i=0;$i< sizeof($propiedad1);$i++){
312.            for( $j=0;$j< sizeof($propiedad2);$j++){
313.                  $distancia=calcularDistancia($propiedad1[$i],
   $propiedad2[$j]);
314.                  $afinidad=calcularAfinidad($propiedad1[$i],
   $propiedad2[$j], $distancia);
315.
316.                        if($afinidad == 1){
317.                        $sum = $sum + $afinidad;
318.                  }else{
319.                        $sumAux = $sumAux + $afinidad;
320.                  }
321.            }
322.            }
323.            if($sum > 0){
324.                  $sumT = $sum/($sum+$sumAux);
325.            }
326.            return $sumT;
327.      }
328.      /*FIN Propiedades*/
```

```
329.      /*Super Clases*/
330.      function similitudSuperClass($claseOnt1, $claseOnt2, $graph1,
   $graph2) {
331.            $superC1 = array();
332.            $superC2 = array();
333.            $simParcial = array();
334.            //$simProp = array();
335.
336.            $pref1 = "PREFIX a: <".$graph1.">";
337.            $pref2 = "PREFIX a: <".$graph2.">";
338.
339.            $sparql = $pref1." SELECT * FROM <".$graph1."> WHERE {
   a:".$claseOnt1." rdfs:subClassOf ?s }";
340.            $result1 = sparql_query( $sparql );
341.            $max_pro1 = sizeof($result1->rows);
342.            for($i = 0; $i < $max_pro1;$i++){
343.                  $posicion_coincidencia = strrpos($result1-
   >rows[$i]["s"]["value"], "nodeID://");
344.                  if ($posicion_coincidencia === false) {
345.                  $pro = generate_atributos($result1-
   >rows[$i]["s"]["value"]);
346.                  array_push($superC1, $pro);
347.            }
348.            }
349.            $sparql = $pref2." SELECT * FROM <".$graph2."> WHERE {
   a:".$claseOnt2." rdfs:subClassOf ?s }";
350.            $result2 = sparql_query( $sparql );
351.            $max_pro1 = sizeof($result2->rows);
352.            for($i = 0; $i < $max_pro1;$i++){
353.                  $posicion_coincidencia = strrpos($result2-
   >rows[$i]["s"]["value"], "nodeID://");
354.                  if ($posicion_coincidencia === false) {
355.                  $pro = generate_atributos($result2-
   >rows[$i]["s"]["value"]);
356.                  array_push($superC2, $pro);
357.            }
358.            }
359.            /*echo "Superclases 1";
360.            print_r($superC1);
361.            echo "Superclases 2";
362.            print_r($superC2);*/
363.      $sum = 0; $sumT= 0; $sumAux = 0; $max1 = sizeof($superC1); $max2 =
   sizeof($superC2);
364.            if($max1 > 0 && $max2 > 0){
365.                  for( $i=0;$i< $max1;$i++){
366.                  for( $j=0;$j< $max2;$j++){
367.                        $distancia=calcularDistancia($superC1[$i],
   $superC2[$j]);
368.                        $afinidad=calcularAfinidad($superC1[$i],
   $superC2[$j], $distancia);
369.                        $afinidad_pro =
   similitudPropiedades($superC1[$i], $superC2[$j], $graph1, $graph2);
370.
371.                        $simPar = ($afinidad + $afinidad_pro)/2;
372.                        array_push($simParcial, $simPar);
373.                  }
```

```
374.      }else{
375.                  $val = 0;
376.            }
377.            return $val;
378.      }
379.
380.
381.      if( !$result ) { print sparql_errno() . ": " . sparql_error().
   "\n"; exit; }
382.
383.      echo json_encode($result);
384.      //return json_encode($result);
```

Graficos.php

```
<?php

$nombre = $_POST['nombre'];
$ubicacion = $_FILES["ubicacion"]["tmp_name"];

$ubicacion = str_replace("\\", "/", $ubicacion);


$first = "C:\Program Files\OpenLink Software\Virtuoso 7.2\bin\isql.exe";
$second = "isql -U dba -P dba -S 1111 EXEC=";
$third =
"DB.DBA.RDF_LOAD_RDFXML_MT(file_to_string_output('".$ubicacion."'),'','".
$nombre."');";
//echo '"'. $first .'" ' . $second.'"' .$third.'"' ;

$out = shell_exec('"'. $first .'" ' . $second.'"' .$third.'"' );

if (strpos($out, 'Done') !== false) {
    //echo 'true';
    $result["resultado"]= 1;
}else{
       //echo "false";
       $result["resultado"]= 0;
}

if( !$result ) { print sparql_errno() . ": " . sparql_error(). "\n";
exit; }

echo json_encode($result);

?>
```

Uploadowl.php

```
<script type="text/javascript"
   src="https://www.gstatic.com/charts/loader.js"></script>

<?php
/*define('__ROOT__', dirname(dirname(__FILE__)));
include(__ROOT__.'/ontoApi/core/examples/tablas.php');*/
?>
<html><!--lang="en"-->

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Analisis Lexico</title>

    <!-- Bootstrap Core CSS -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom CSS -->
    <link href="css/sb-admin.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <link href="font-awesome/css/font-awesome.min.css" rel="stylesheet"
   type="text/css">
<script>


    function upload() {
        var xmlhttp = new XMLHttpRequest();
        var url = "core/examples/graficos.php";
        var data = new FormData(document.getElementById("form_ingreso"));
        xmlhttp.onreadystatechange=function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                 var array = JSON.parse(xmlhttp.responseText);

                // console.log(array);

                if(array.resultado == 1){
                    //window.open("http://127.0.0.1/sc/index.php");
                    //document.location = "index.php";
                    alert("Archivo almacenado");
                }else{
                    alert("Error en la carga del archivo");
                }
            }
}
        xmlhttp.open("POST", url, true);
        xmlhttp.send(data);
    }
```

```
95. </script>
96. </head>
97.
98. <body>
99.
100.          <div id="wrapper">
101.
102.              <!-- Navigation -->
103.              <nav class="navbar navbar-inverse navbar-fixed-top"
    role="navigation">
104.                  <!-- Brand and toggle get grouped for better mobile
    display -->
105.                  <div class="navbar-header">
106.                      <button type="button" class="navbar-toggle" data-
    toggle="collapse" data-target=".navbar-ex1-collapse">
107.                          <span class="sr-only">Toggle navigation</span>
108.                          <span class="icon-bar"></span>
109.                          <span class="icon-bar"></span>
110.                          <span class="icon-bar"></span>
111.                      </button>
112.                      <a class="navbar-brand"
    href="index.php">ANALIZADOR DE ONTOLOGÍAS</a>
113.                      <span>|</span>
114.                      <a class="navbar-brand" href="index.php">CARGAR
    ONTOLOGÍAS</a>
115.                  </div>
116.      </div>
117.                  <!-- /.navbar-collapse -->
118.              </nav>
119.
120.              <div id="page-wrapper">
121.                  <div class="container-fluid">
122.
123.                      <!-- Page Heading -->
124.                      <div class="row">
125.                          <div class="col-lg-12">
126.                              <h1 class="page-header">
127.                                  Técnicas de Minería Ontológica <small>
    (Sparqllib)</small>
128.                              </h1>
129.                          </div>
130.                      </div>
131.      <div class="row">
132.                          <form id="form_ingreso" name="form_ingreso"
    href="javascript:;" action="openontology();return true;">
133.                              <div class="col-lg-10">
134.                                  <div class="panel panel-primary
    class">
135.                                      <div class="panel-heading">
136.                                          <h3 class="panel-
    title">SELECCIONAR ONTOLOGÍA <!-- <small> Grafo 1</small>
    style="width:100%;"--></h3>
137.                                      </div>
138.      <div class="panel-body">
139.                                          <label>Graph <input
    name="nombre" type="text" id="nombre" /></label>
```

```
142.      </div>
143.                              </div>
144.
145.                              <div class="col-lg-6">
146.                                  <label><input type="button"
   href="javascript:;" onclick="upload();return false;" class="btn btn-
   success" value="Subir"/></label>
147.                              </div>
148.
149.                          </form>
150.                      </div>
151.      </div>
152.                  <!-- /.container-fluid -->
153.
154.              </div>
155.              <!-- /#page-wrapper -->
156.
157.          </div>
158.          <!-- /#wrapper -->
159.
160.          <!-- jQuery -->
161.          <script src="js/jquery.js"></script>
162.
163.          <!-- Bootstrap Core JavaScript -->
164.          <script src="js/bootstrap.min.js"></script>
165.
166.      </body>
167.
168.      </html>
```