Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Proyecto de Integración:

Sistema de agrupamiento de servicios web semánticos utilizando un algoritmo bioinspirado

Saúl Eduardo Santillán Pérez

Matricula: 208367814

Asesores:
Maricela Claudia Bravo Contreras
Profesora Asociada, Departamento de Sistemas

Roman Anselmo Mora Gutiérrez
Profesor Asociado, Departamento de Sistemas

Trimestre 2014 Primavera de Agosto de 2014

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

Yo, Roman Anselmo Mora Gutiérrez, *declaro que aprobé el contenido del presente Reporte de Proyecto de Integración* y *doy mi autorización para su* publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Saúl Eduardo Santillán Pérez, *doy mi autorización* a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Contenido

Índice de Figuras

Resumen

El sistema realizado consiste de un clasificador de servicios web semánticos, utilizando como motor de clasificación el algoritmo ACO[1] (Ant Colony Optimization, en sus siglas en ingles).

En este trabajo analizamos el desempeño del algoritmo ACO modificado para clasificar una colección de servicios web semánticos recuperados de una ontología local, para que un usuario programador realice la búsqueda de una manera más específica y con un tiempo de ejecución más rápido, ya que actualmente éste proceso de búsqueda los resultados son demasiados, muy poco específicos y por lo tanto el tiempo es mucho mayor.

La clasificación se realizara utilizando medidas de similitud semánticas, se compararan semánticamente NxN servicios recuperados de la ontología local, se obtiene de cada métrica de similitud un valor único, que es el valor de similitud de cada servicio entre todos los demás, estos datos serán colocados en una matriz de similitud, para después ser procesada por el ACO y obtener una clasificación de los servicios web semánticos.

[1] Algoritmo de la colonia de hormigas.

1. Introducción

En la actualidad existen gran cantidad de algoritmos biológicos, que han representado fuentes de inspiración para la informática, un ejemplo es el algoritmo de la colonia de hormigas, ya que se basa en la forma en que estos insectos encuentran la ruta más corta desde su nido a su fuente de alimentación [1].

El Algoritmo Ant Colony Optimization, es una heurística[2] que se basa en comportamiento de las hormigas en la naturaleza por intentar encontrar una ruta más corta entre su nido y una fuente de alimento [2].

Las hormigas se comunican entre sí por medio de feromonas[3], para dejar un rastro y crear un camino. Las feromonas proporcionan información acerca de qué camino debe ser seguido. Cuanto mayor sea el número de hormigas que traza un camino dado, mayor es el atractivo para seguir este camino por otras hormigas mediante el depósito de su propia feromona (ver Figura1).

El primer sistema de ACO fue presentado por Marco Dorigo en su Tesis (1992), y fue llamado Ant System (AS). AS es el resultado de una investigación sobre la inteligencia computacional enfoques de optimización combinatoria que Dorigo llevó a cabo en el Politécnico di Milano en colaboración con Alberto Colorni y Vittorio Maniezzo. AS se aplicó inicialmente al problema del viajante de comercio, y para el problema de asignación cuadrática.fue llamado Ant System (AS). AS es el resultado de una investigación sobre la inteligencia computacional enfoques de optimización combinatoria que Dorigo llevó a cabo en el Politécnico di Milano en colaboración con Alberto Colorniy Vittorio Maniezzo. AS se aplicó inicialmente al problema del viajante de comercio, y para el problema de asignación cuadrática [3].

[2] La heurística se refiere a las técnicas basadas en la experiencia para resolución de problemas, el aprendizaje y el descubrimiento.

[3] Las feromonas son sustancias químicas secretadas por los seres vivos con el fin de provocar comportamientos específicos en otros individuos de la misma especie.

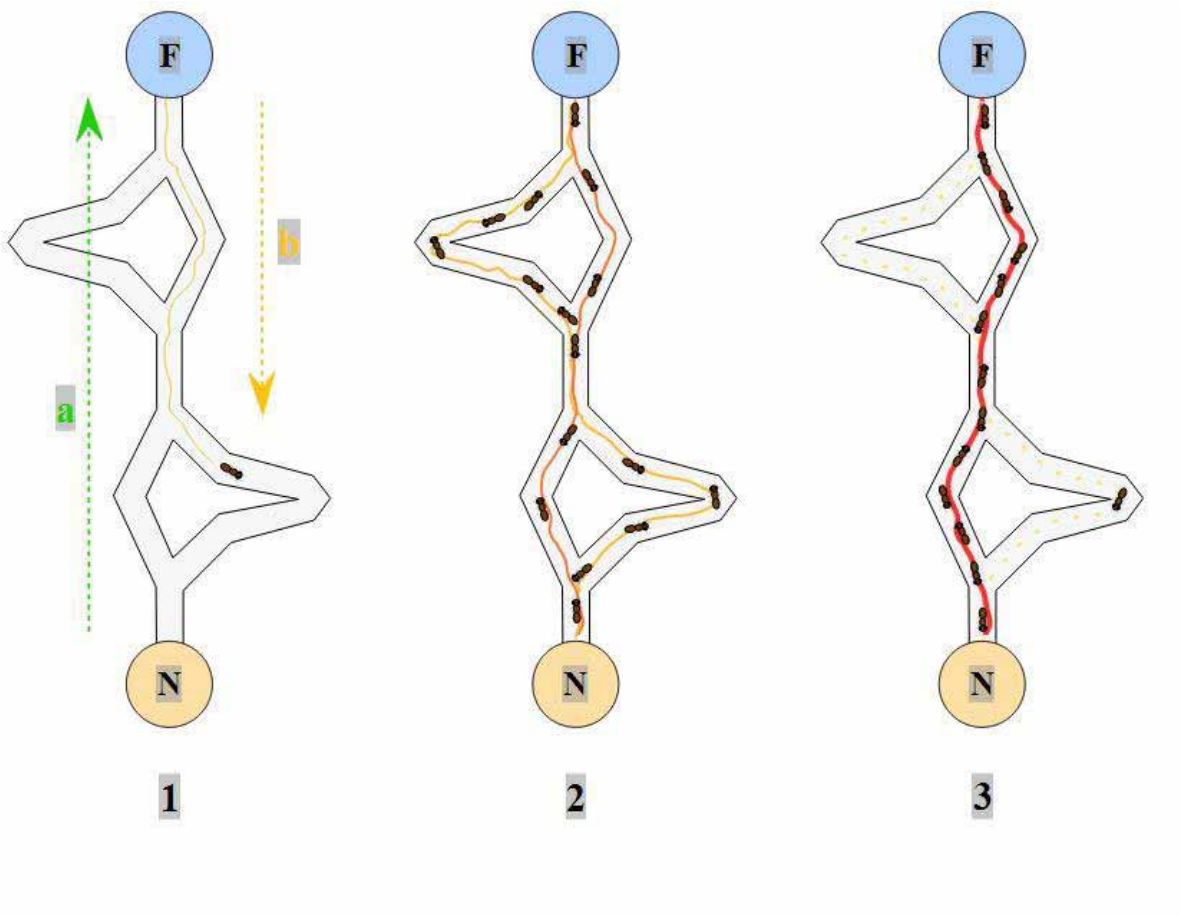

Figura 1.- Comportamiento del Ant Colony Optimization[4]

1. La primera hormiga vaga al azar hasta que encuentra la comida fuente (F), a continuación, vuelve al nido (N), por la que se una rastro de feromona
2. Otras hormigas siguen uno de los caminos al azar, también por el que se rastros de feromonas. Dado que las hormigas en la disposición del camino más corto feromona rutas más rápido, este camino se refuerza con más feromona, lo que es más apelando a las hormigas futuras.
3. Las hormigas se vuelven cada vez más propensas a seguir el camino más corto ya que se refuerza constantemente con una mayor cantidad de feromonas. La feromona que se encuentra en caminos más largos tiende a evaporarse, evitando ser atractivo para la hormiga.

[4]http://es.wikipedia.org/wiki/Algoritmo_de_la_colonia_de_hormigas#mediaviewer/Archivo:Aco_branches.svg

En los algoritmos ACO(ver figura 2), las hormigas artificiales construyen una solución para Problemas de Optimización Combinatoria (POC[5]) atravesando un grafo llamado grafo de construcción, GC(V,E). El grafo de construcción (totalmente conectado) consiste de un conjunto de vértices V y un conjunto de arcos E. El conjunto de componentes C puede ser asociado con el conjunto de vértices V, o con el conjunto de arcos E.

Las hormigas se mueven de un vértice a otro vértice a lo largo de los arcos del grafo, construyendo incrementalmente una solución parcial. Además, las hormigas depositan una cierta cantidad de feromona sobre las componentes, es decir, en los vértices o en los arcos que atraviesan. La cantidad de feromona $\Delta\tau$ depositada puede depender de la calidad de la solución encontrada. Las hormigas siguientes utilizan la información de la feromona como una guía hacia regiones más prometedoras del espacio de búsqueda.

El leguaje de descripción de servicios web (WSDL[6] por sus siglas en inglés) es un formato XML[7] que se utiliza para describir servicios web (ver Figura 3).

OWL-S es una ontología que describe y extiende el concepto de servicio web, es decir, agregándole recursos para relacionar los datos entre ellos, esto permite que puedan ser utilizados de forma autónoma por un sistema de información con acceso a la web (ver figura 4).

[5] Poblem Optimization Combinatory

[6] WSDL son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web . La versión 1.0 fue la primera recomendación por parte del W3C y la versión 1.1 no alcanzó nunca tal estatus. La versión 2.0 se convirtió en la recomendación actual por parte de dicha entidad.

[7] Por su siglas en inglés de eXtensible Markup Language , es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

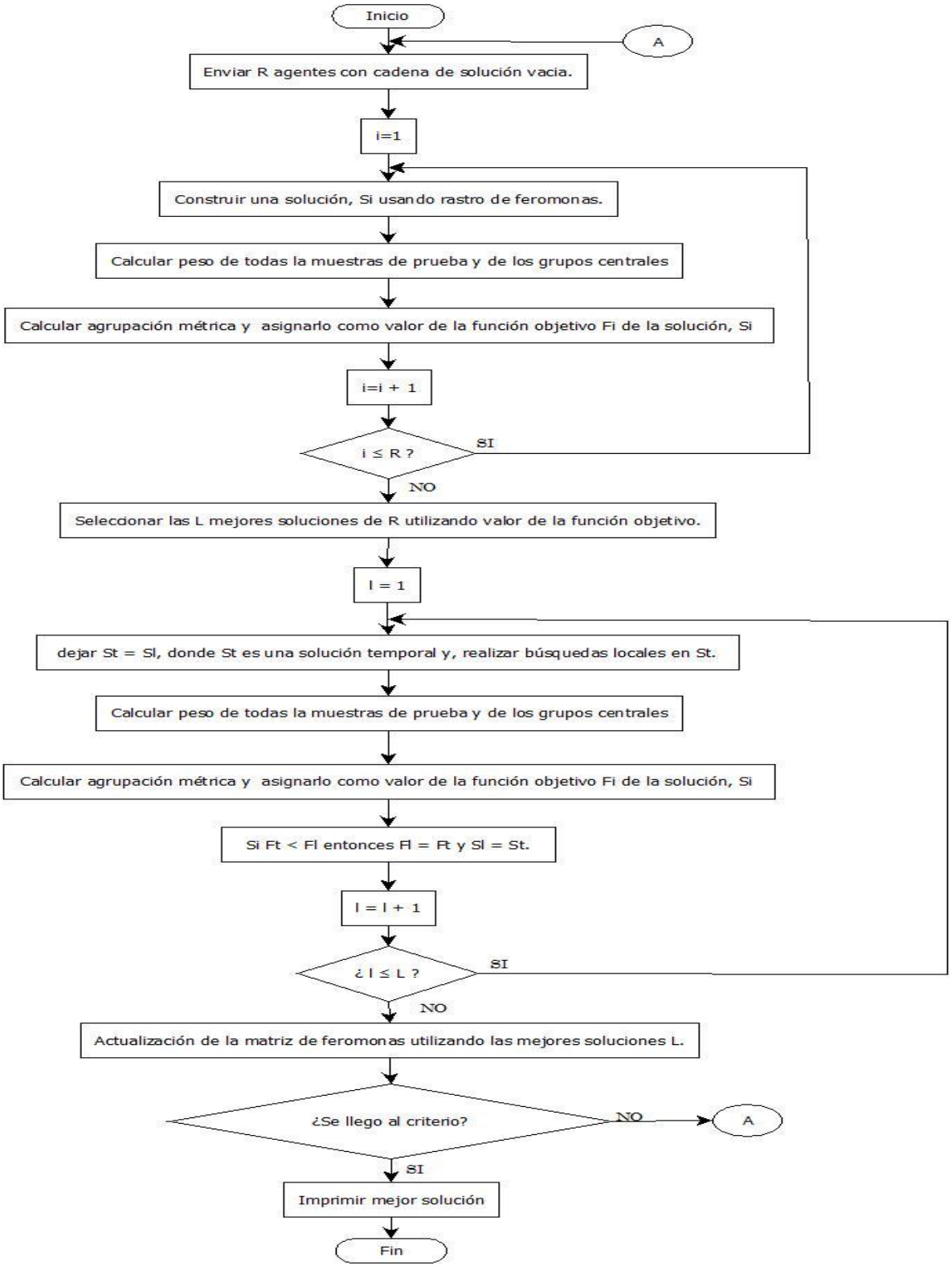

Figura 2.-Algorimo General de ACO

La clasificación de los servicios se basa en la similitud semántica entre las operaciones de los servicios web, para obtener la similitud se utilizan las métricas Wu-Palmer, Lin, Lesk y Path, estas métricas son WordNet's[8] de similitud establecidas en una base de datos léxica, cada una de ellas utiliza una ecuación en particular para calcular la similitud semántica de dos palabras [4].

El objetivo de la clasificación de los servicios web semánticos es que su búsqueda sea más específica, sin tener que consultar todos los resultados, así como minimizar el tiempo en el proceso de búsqueda del usuario programador.

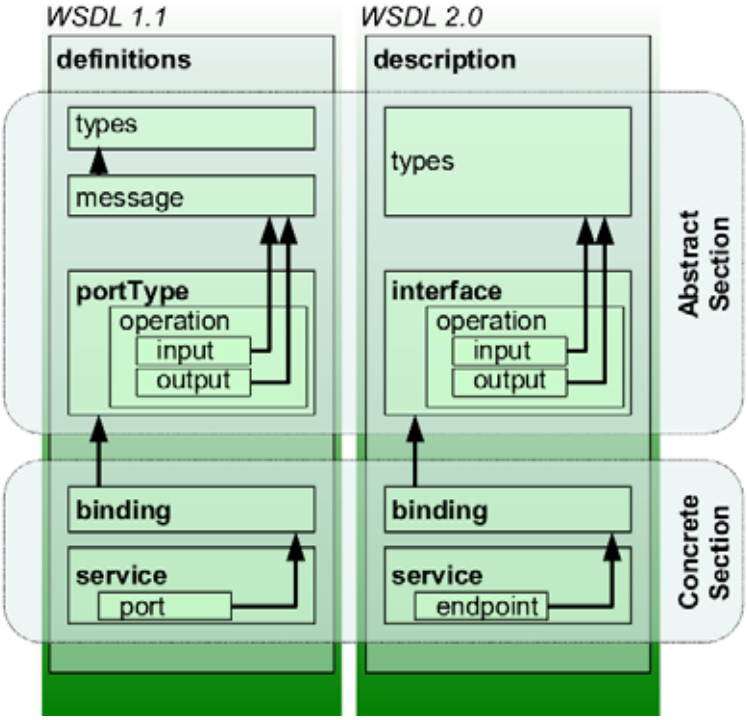

Figura 3.-Arquitectura básica de un servicio web

[8] medidas de similitud y relación semántica basada en la información encontrada en la base de datos léxica

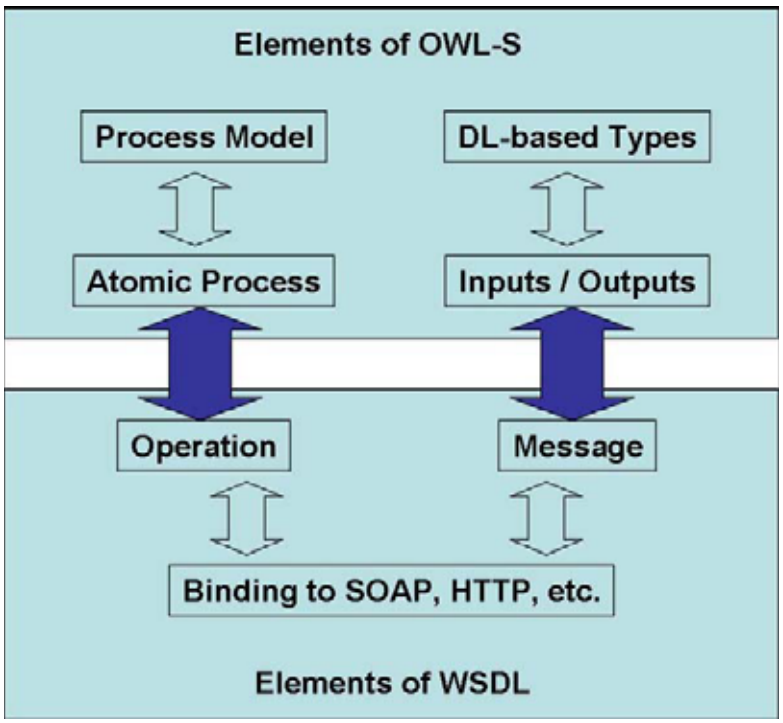

Figura 4.- Arquitectura de un servicio web Semántico (OWL-S)

2. Antecedentes

2.1. Proyectos Terminales

- Clasificación de servicios web mediante ontologías [5]. Éste proyecto es similar ya que trata de realizar un agrupamiento de servicios web mediante ontologías, sin embargo la clasificación de estos servicios web se realiza mediante mapas auto organizados.

- Extracción automatizada y representación de servicios web mediante ontologías [6]. Éste proyecto es similar ya que se basa en servicios web contenidos en una ontología, pero no atiende la agrupación de los servicios web.

- Sistema clasificador de ontologías mediante métodos de máquinas de soporte vertical [7]. Este proyecto es similar ya que realiza clasificación de ontologías. Sin embargo, no hace referencia a servicios web además no utiliza un algoritmo bioinspirado para realizar el agrupamiento.

2.2. Artículos

- Sistema de recuperación de servicios web basado en un modelo de taxonómica [8]. Este sistema es muy parecido ya que hace un agrupamiento de servicios web mediante una discriminación específica. Sin embargo esto lo realiza utilizando una ontología, mi propuesta utilizara la discriminación ya contenida dentro de las ontologías.

- Un algoritmo de Agrupamiento basado en ACO [9]. Este artículo es muy similar ya que se basa en la heurística de ACO para el agrupamiento, pero el sistema ocupa el algoritmo para agrupar datos, por lo cual está orientado a la minería de datos. En este trabajo implementaremos un ACO orientado a la agrupación de servicios web semánticos.

2.3. Tesis

- Arquitectura de servicios web semánticos sensible al contexto para dispositivos móviles [10]. Este proyecto es muy similar ya que se basa en los servicios web. Sin embargo no hace uso del agrupamiento de servicios web además que es utilizado por dispositivos móviles.

3.Justificación

Hoy en día existen gran cantidad de buscadores de servicios web semánticos, los cuales ya cuentan con una clasificación de estos servicios. Sin embargo, lo realizan de una manera generalizada, por lo que la agrupación es muy poco específica, esto provoca una infinidad de resultados, por lo consiguiente provocan que el usuario programador ejecute su búsqueda con un mayor tiempo.

Es por eso que con este proyecto ofreceremos al usuario programador una búsqueda más específica, ya que la clasificación de los servicios web semánticos se realiza con relaciones semánticas entre ellos, gracias a estas relaciones brindaremos al usuario una búsqueda más objetiva que se adapte a sus necesidades, así como posibles resultados que se relacionen con su búsqueda más específica, obteniendo mejora en el tiempo de buscada de servicios web por parte del usuario programador.

El agrupamiento lo realizamos mediante el uso del algoritmo heurístico inspirado en la colonia de hormigas (ACO), ya que el comportamiento de los algoritmos de agrupamiento actuales es generalizado, con el ACO esperamos obtener mejores resultados en el comportamiento de agrupamiento , es por eso que la principal motivación para realizar este proyecto son los buenos

resultados que los algoritmos heurísticos bioinspirados han mostrado en otras aplicaciones, así como la innovación de utilizarlos en la clasificación de servicios web semánticos, es por eso que se utilizara el ACO para realizar esta clasificación, ya que ha mostrado buenos resultados en aplicaciones similares, como por ejemplo en minería de datos y en proyectos relacionados con servicios web [11].

4. Objetivos

Diseñar e implementar un algoritmo bioinspirado, para el agrupamiento automatizado de servicios web semánticos.

4.1. Objetivos Específicos

- Diseñar e implementar un módulo de conexión a las ontologías de servicios web para recuperación de los servicios.
- Implementar un algoritmo de agrupamiento inspirado en Ant Colony Optimization.
- Evaluar el agrupamiento de los servicios.

5. Desarrollo del Proyecto

En este proyecto se diseñó e implemento un sistema que agrupa servicios web semánticos, estos servicios serán extraídos de un directorio local para posteriormente realizar el proceso de agrupamiento mediante el algoritmo de la colonia de hormigas. Al concluir el proceso de agrupamiento, el sistema proporciona la visualización de los resultados. Éste proyecto consta de los siguientes módulos (ver Figura 3):

- Recuperación de Datos
  Este módulo realiza una conexión con el directorio local, para recuperar los servicios web semánticos.

- Módulo de agrupamiento

Este módulo realiza el agrupamiento de los servicios web semánticos, utilizando el ACO modificado para conseguir esta acción.

- Módulo de Evaluación de Resultados por Expertos
  En este apartado del proyecto se procede a evaluar los resultados obtenidos por el sistema, teniendo en cuenta ya su calibración y después de realizar las corridas para el análisis de los resultados.

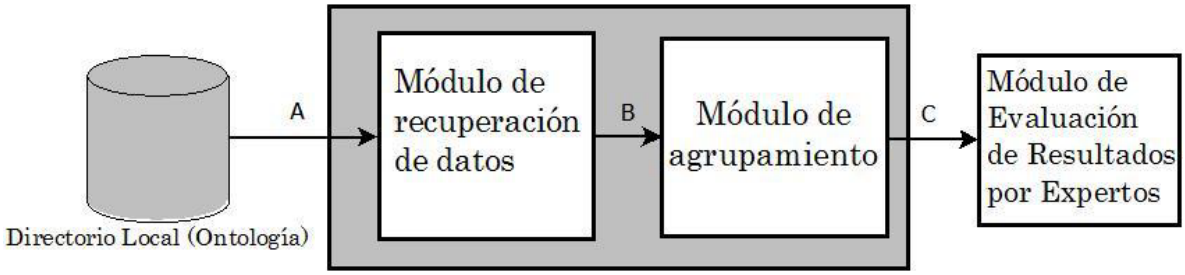

Figura 5.-Arquitectura del sistema

5.1. Módulo de recuperación de datos

Este módulo permite recuperar los servicios web semánticos con extensión OWL-S desde una Ontología (Repositorio Local) poblado con anterioridad (Colección de servicios). Se realiza una conexión a la Ontología por medio de la API-OWL[9], para extraer el nombre Operación y los parámetros (entrada y salida) de los servicios web semánticos, estos datos se guardan en Objetos tipo Operación, para poder manipularlos posteriormente(ver Figura 6).

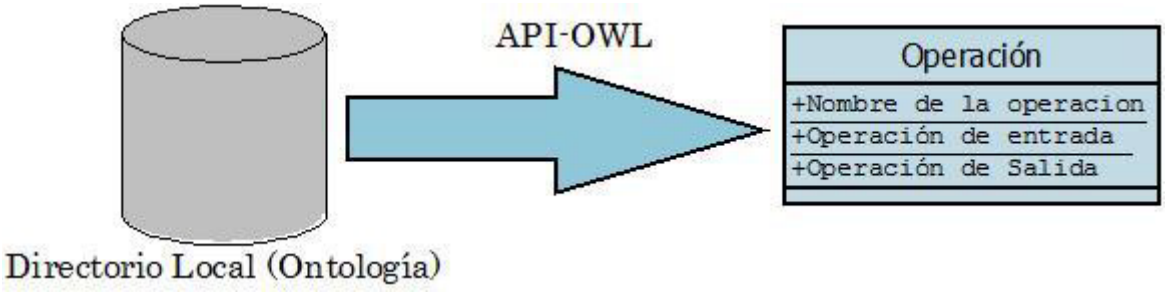

Figura 6.-Arquitectura Módulo de Recuperación de datos

[9] API-OWL API de java para crear, manipular y serialización ontologías

El siguiente paso es comparar semánticamente las operaciones de entrada y de salida de cada servicio con los demás, es decir, una comparación NXN servicios web, utilizando las siguientes métricas de similitud semántica:

- Lin
- Wu-Palmer
- Path
- Lesk

Las métricas de similitud regresan un valor entre 0 y 1 donde:

- 0 si el parentesco semánticamente entre los servicios es igual
  Y
- 1 si no hay nada de parentesco semántico entre los servicios.

El valor de similitud de cada métrica es una matriz, a la cual se le saca el promedio de la semántica entre los dos servicios comparados, es decir, el valor de la similitud se obtiene sumando el valor semántico de cada palabra contenida en cada una de las operaciones y dividido entre el número de palabras por las que está constituida las dos operaciones

El resultado de esta operación es una matriz cuadrada de tamaño N que contiene los valores de similitud semántica de los NxN servicios web recuperados de la ontología, por cada métrica de similitud semántica se obtiene una matriz y se creara una matriz adicional que alberga el promedio de los valores de las cuatro métricas anteriores, por lo tanto, en este proceso se producirán cinco matrices:

- LinSimilarity
- WuPalmerSimilarity
- PathSimilarity
- LeskSimilarity
- PromSimilarity[10]

El código de este módulo se puede consultar en el Anexo B.

[10] Esta Matriz contiene el promedio de los valores de la métrica Lin, Lesk, Path y Wu-Palmer.

5.2. Módulo de agrupamiento

Una vez realizado el proceso de recuperación de los servicios web y de calcular las matrices de similitud a partir de la semántica de sus operaciones, procedemos a realizar la clasificación. Una de las matrices calculadas es el parámetro de entrada para el algoritmo de agrupamiento inspirado en la colonia de hormigas (ver Figura 7), dicho algoritmo procesa iterativamente, es decir, trata de llegar a la mejor solución mediante aproximaciones sucesivas, cada una de las hormigas virtuales que usemos es una posible solución, cada una de ella devuelve una clasificación. En este proyecto utilizamos un algoritmo ACO modificado que es una variante del ACO, las diferencia entre ellos se pueden observar en la Tabla 1.

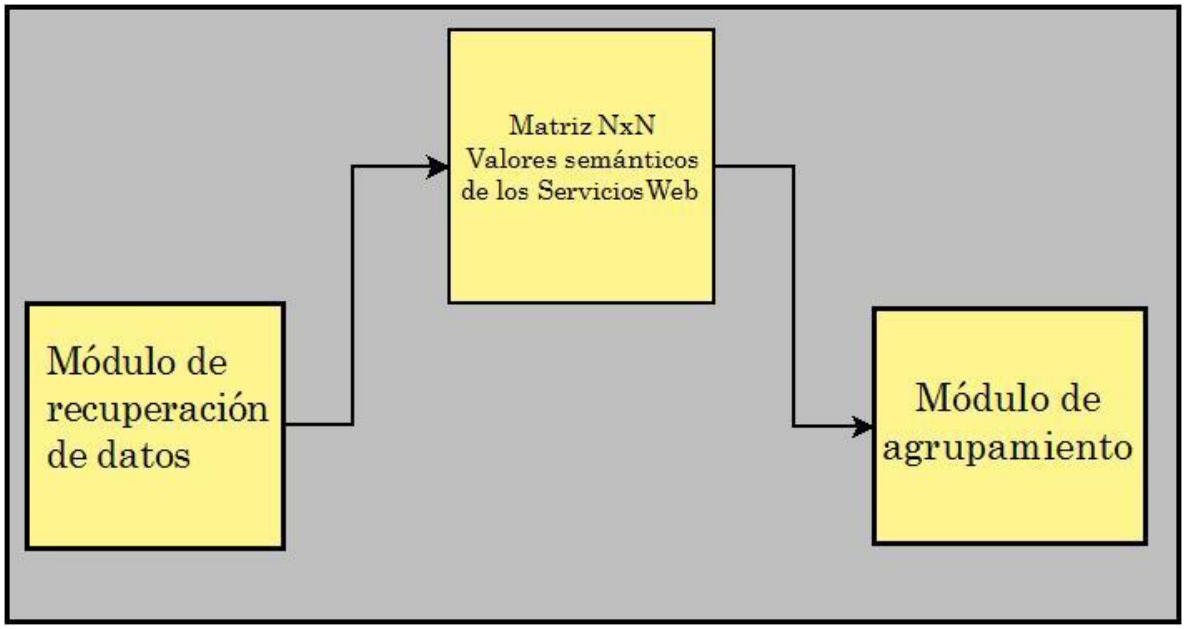

Figura 7.- Entrada de datos para el Módulo de agrupamiento

El algoritmo ACO que utilizamos es una variante, ya que este algoritmo emplea una decisión por consenso para obtener la mejor solución, es decir, mediante un proceso de elección se busca no solamente el acuerdo de la mayoría de los participantes, sino que también se persigue el objetivo a resolver.

El primer paso del algoritmo es cargar los componentes y parámetros para que afectaran la el comportamiento del mismo, como por ejemplo, los parámetros α, β y ρ, que permiten ajustar el costo de los valores de los

rastros de feromonas y de la información heurística, después se lee la matriz de similitud semántica, posteriormente crea un arreglo de distancias medias, así como inicializa al conjunto de soluciones que en nuestro caso es el número de hormigas artificiales que ejecutaran el algoritmo.

Cada hormiga en forma concurrente, independiente, construye socialmente una solución mediante la incorporación de información a parámetros sobre una solución parcial. La incorporación de la información se realiza utilizando datos probabilísticos como por ejemplo la varianza, así contribuir a la experiencia adquirida en la búsqueda de la mejor solución.

ACO	*ACO modificado*
• La mejor solución se obtiene conforme se va actualizando la matriz de feromona en cada iteración del algoritmo, la feromona contribuye a la experiencia de las otras hormigas a seguir el camino a la mejor solución en cada iteración. • Solo utiliza un depósito de feromona para encontrar la mejor solución de acuerdo al parámetro de convergencia.	• Este algoritmo es un hibrido, ya que la mejor solución se va construyendo conforme el algoritmo va iterando, utiliza un Proceso de Elección por consenso para obtener la mejor solución, todos los miembros del conjunto de soluciones tiene que estar de acuerdo en que ese miembro es el mejor de todos para perseguir la solución. • Utiliza dos depósitos de feromonas, la utilidad de las feromonas se expresa a continuación: ▪ La primera feromona está ligada al número de grupos que se van a formar. ▪ La segunda feromona proporciona la información de que agrupación se coloca cada servicio.

Tabla 1.-Comparativa ACO vs ACO Modificado

Para contribuir a la experiencia se utiliza una matriz de feromona para colocar un servicio en un Cluster[11] determinado, y otra matriz de

[11] Se refiere a agrupamiento en ingles.

feromonas que está ligada al número de Cluster's que se van formando; estas matrices se utilizan como memoria que almacena el rastro depositado por las hormigas en la construcción de la mejor solución con cada iteración. Cuando todas las hormigas han construido su solución se actualiza el rastro de feromona depositado en las matrices. El valor de los rastros almacenados en la matriz pueden incrementarse por el depósito o decrementarse por la evaporación.

El depósito de feromona se realiza actualizando las matrices usadas en la construcción de buenas soluciones, aumentando su probabilidad de ser seleccionadas en el futuro para intensificar la búsqueda en regiones próximas a soluciones. De manera complementaria, la evaporación de feromona evita la convergencia prematura a regiones no ´optimas, fomentando la exploración del espacio de búsqueda por la utilización de la información heurística, esto se realiza utilizando funciones para minimizar la varianza en términos de la distancia entre los nodos que en nuestro caso seria los Clúster's y en número de elementos contenidos en cada uno de estos agrupamientos.

En cada iteración del algoritmo se obtiene a la mejor hormiga de la colonia, esta hormiga es que mejor proporciona una posible solución, a partir de ella se actualizan las matrices de feromonas. La mejor hormiga se localiza ordenando las hormigas acuerdo a su costo y a las soluciones construidas, pondera su contribución en la actualización del rastro de acuerdo a su posición [12].Una vez que finalizado este proceso iterativo, se regresa la mejor solución con la mejor hormiga, se muestran los resultados para poder analizarlos.

Este módulo de agrupamiento será implementado en MATLAB 12b[12](ver Anexo C).

[12] MATLAB es el lenguaje de alto nivel y un entorno interactivo utilizado por millones de ingenieros y científicos de todo el mundo. Se le permite explorar y visualizar las ideas y colaborar en todas las disciplinas, incluyendo señales y procesamiento de imágenes, comunicaciones, sistemas de control, y las finanzas computacionales.

5.3. Módulo de Evaluación de Resultados por Expertos

En este módulo se analiza los resultados obtenidos después del agrupamiento, teniendo en cuenta los factores como la colección de servicios se va a clasificar y la calibración previa del ACO-modificado. El análisis de los datos se realiza por un experto servicios web semánticos.

6. Metodología

La entrada de nuestro sistema es una ontología local, la cual contiene la colección de servicios web semánticos, que posteriormente se clasificaran (ver Figura 5).

Una vez creado nuestro proyecto en Java debemos agregar las librerías Ws4j-1.0-1.jar (esta librería contiene las métricas semánticas) y Owl-API.jar (esencial para realizar la conexión a la ontología) en el caso de que no estén agregadas. Posteriormente, debemos colocar la ruta del directorio local que contiene a nuestra ontología en el parámetro *localDir*, de la misma forma colocamos la ruta completa de nuestra ontología en el parámetro *ontologyFile* de nuestra clase *mainTester* (ver Anexo B), como se muestra en la Figura 8.

```
public static void main(String[] args)
{
    //primer paso Cargar la ontologia
     String localDir = " /home/saul/NetBeansProjects/PTHormiga/Ontologias";
     String ontologyFile = "/home/saul/NetBeansProjects/PTHormiga//Ontologias/Prueba.owl";
    String ontologyIRI = "http://www.semanticweb.org/ontologies/2012/8/templateWSDL10.owl";
```

Figura 8.- Ruta para cargar la Ontología de servicios

Una vez realizado las rutas donde se encuentra la ontología procedemos a ejecutar el la clase *mainTester*, esta clase se encarga de cargar la ontología para realizar la extracción de las operaciones de cada servicio web semántico, por cada servicio se crea un objeto llamado *operación* que contiene el nombre de la operación, sus mensajes de entrada y de salida. Una vez concluida la extracción de los datos, se procede a generar las matrices correspondientes de cada métrica de similitud ya mencionadas en el apartado 5.1 del presente documento.

Las matrices de similitud generadas se localizan en el directorio local donde este alojado nuestro proyecto, en formato (.txt, ver Figura 9). Adicionalmente se genera un archivo llamado “Operaciones_servicios”, en

este archivo contiene las operaciones que fueron comparadas semánticamente.

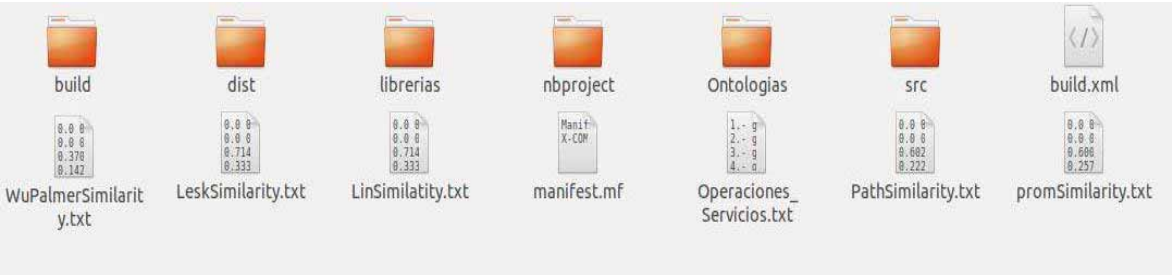

Figura 9.- Matrices generadas en el Directorio Local del Proyecto Java

El siguiente paso es copiar una matriz de similitud generada al directorio local de nuestro proyecto en Matlab 12b (ver Figura 10), a continuación colocamos el nombre de la matriz de similitud semántica que copiamos, en el parámetro *importdata* de nuestra función *ejecute.m* (ver Anexo C), una vez realizado esto procedemos a ejecutar nuestro función *ejecute.m*, esta función se encarga de clasificar los servicios web semánticos, a partir de los datos contenidos en la matriz de similitud semántica.

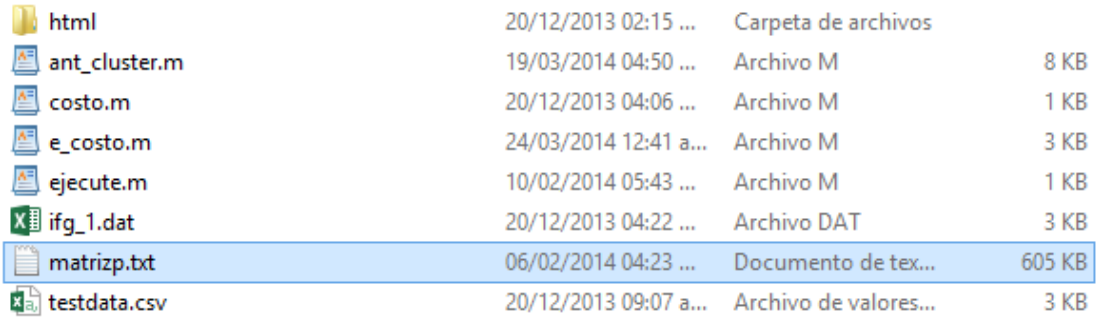

Figura 10.- Directorio local del Proyecto Matlab

Los resultados después de haber corrido la función *ejecute.m* serán mostrados en una hoja de cálculo llamada testdata.csv localizada en el directorio local de nuestro proyecto en Matlab. En esta hoja de cálculo contiene:

- En que clúster coloco a cada servicio.

- El número de clúster que se generaron.
- El número de clúster con un solo elemento.
- El Error promedio
- La Varianza

6.1. Metodología de calibración

El algoritmo se calibro utilizando el método de fuerza bruta, este método consiste en fijar un periodo de tiempo utilizando una configuración inicial específica, con configuración se ejecuta el algoritmo y se observan los resultados.

Posteriormente se realiza modificación en la configuración en uno de los parámetros y se vuelve a ejecutar el algoritmo, se observan los resultados para observar si se mejoran los resultados obtenidos previamente, si se mejoraron los resultados esta configuración se toma como la nueva configuración inicial y se repite el procedimiento, en caso contrario, la configuración se rechaza y se modifica otro parámetro de la configuración inicial. El método se realiza hasta satisfacer el periodo de tiempo establecido al principio.

Para la calibración en este proyecto se fijó el periodo de tiempo de una semana, para realizar las ejecuciones del algoritmo, iniciando con una configuración inicial de los parámetros (ver apartado 8.1) y posteriormente en base a os resultados expuestos por dicha configuración, modificamos algunos de estos parámetros para observar si existe mejoría con la nueva configuración.

7. Implementación

Todos los recursos software son de libre distribución. La mayoría del sistema esta implementado en lenguaje de programación JAVA.

7.1. Tecnologías de implementación

7.1.1. IDE Netbeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

7.1.2. API-OWL

La API-OWL es una API de java para crear, manipular y serialización OWL Ontologies, es una Api de código abierto

7.1.3. MATLAB

Es un lenguaje de alto nivel y un entorno interactivo utilizado por millones de ingenieros y científicos de todo el mundo. Se le permite explorar y visualizar las ideas y colaborar en todas las disciplinas, incluyendo señales y procesamiento de imágenes, comunicaciones, sistemas de control, diseño de algoritmos y las finanzas computacionales.

El módulo de recuperación de datos fue desarrollado en lenguaje Java, utilizando la herramienta API-OWL para poder manipular la ontología y recuperar las operaciones de los servicios web semánticos (ver Anexo B).

El algoritmo ACO para el módulo de agrupamiento se encuentra diseñado e implementado en MATLAB, ya que este lenguaje de alto nivel proporciona un lenguaje de programación propio y herramientas de desarrollo de alto nivel que le permiten desarrollar y analizar algoritmos y aplicaciones de forma rápida, además proporciona opciones más óptimas para el manejo, creación e interacción de matrices (Ver Anexo C).

8. Experimentación y Resultados

8.1. Calibración del algoritmo ACO

Una metaheurística cuenta con parámetros que deben ser debidamente seleccionados y ajustados con vistas a lograr resultados que permitan aprovechar al máximo las ventajas del algoritmo.

Para ello realizamos la calibración de este algoritmo donde se ejecutaron pruebas con una matriz de entrada que junta el promedio de las 4 meticas de similitud llamada "PromSimilarity", esta matriz se obtuvo a través de una ontología de servicios web semánticos llamada Prueba, dicha ontología se pobló con una colección de 41 servicios web con el modelo wsdl, estos servicios ya se sabe que están clasificados en 6 categorías.

Los parámetros que se utilizan para calibrar el algoritmo son los siguientes:

- n_ant → número de hormigas virtuales que ejecutaran el algoritmo.
- q_0 → Parámetro del de balance. debe tener un valor de 0 a 1.
- alpha (α) → Parámetro que permiten ajustar el costo

- beta (β)→ Parámetro que permiten ajustar el costo, debe tener un valor de 0 a 1
- ro (ρ) → Parámetro que permite ajustar los centroides, debe tener un valor de 0 a 1
- distance_max → distancia máxima entre los centroides de cada cluster.

Estos parámetros se pueden observar en el Anexo C clase e*jecute*.m. Se calibro el algoritmo para que en cada prueba realizara veinte ejecuciones al algoritmo ACO modificado, el algoritmo se calibro con el método de fuerza bruta (descrito en el apartado 6.1 de este documento) con un periodo de tiempo de 1 semana. Se obtuvieron los resultados mostrados en la Tabla 2.

Parámetros de calibración	Ejecución Número										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
n_ant=	15	15	15	15	10	16	15	15	15	15	10
q_0=	0.3	0.35	0.35	0.3	0.3	0.3	0.35	0.35	0.25	0.35	0.35
α =	2	2	2	2	3	2	2.6	2.6	2	2	2
β =	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.3
ρ =	0.4	0.4	0.37	0.4	0.3	0.4	0.4	0.39	0.39	0.38	0.4
distance_max=	0.5	0.5	0.05	0.05	0.5	0.5	0.5	0.5	0.5	0.05	0.5
N. iteraciones correctas con seis Cluster´s =	5	7	4	4	2	4	6	9	8	6	6

Tabla 2.- Resultados de la Calibración.

Como ya sabemos que en esta colección de 41 servicios web semánticos están divididos en seis categorías, observamos qué ejecución del algoritmo devuelve un total de 6 Clusters. Como podemos observar (ver Tabla 2) la ejecución numero 8 es la que devuelve el mayor número de iteraciones con 6 clusters, por lo que la configuración de los parámetros utilizada en esta prueba es la indicada para que el algoritmo ACO implementado en este proyecto obtenga los resultados esperados, es la siguiente:

- n_ant = 15
- q_0 = 0.35
- α = 2.6
- β =0.25
- ρ =0.39
- distance_max = 0.5

Con esta configuración de parámetros el algoritmo obtuvo el número máximo de resultados correctos, con un total de nueve de las veinte ejecuciones que se realizaron con esta configuración.

8.2. Ejecución del algoritmo con métricas de similitud.

Después de haber realizado la calibración del algoritmo ACO, procedemos a obtener las matrices de similitud semántica de las cuatro métricas mencionada anteriormente (ver apartado 5.1), cada una de las métricas da un valor semántico diferente, es por eso que realizamos la ejecución del algoritmo a cada una de ellas, y así concluir cuál de estas métricas es la mejor para la clasificación de los servicios web semánticos. Utilizamos la misma colección de 41 servicios llamada Prueba, ya que sabemos que están divididos en 6 categorías, por lo tanto podemos comparar de acuerdo a los resultados que métrica de similitud semántica arroja el mayor resultado igual a 6 en cada una de sus 20 ejecuciones.

Una vez realizada la extracción de los datos por medio del módulo de recuperación de datos, se obtienen una matriz cuadrada por cada métrica, en este caso se obtiene una matriz cuadrada 41 x 41, a continuación se muestran los resultados obtenidos al correr el algoritmo con cada una de las métricas de similitud semánticas, en cada una de las tablas se muestra el número de ejecución del algoritmo, seguido del número de categorías o Clusters encontrados, así como el número de Clusters que contienen un solo elemento solo elemento:

- Resultados Promedio de las Métricas

Ejecuciones del Algoritmo ACO	Número de Clústeres	Número de clústeres con un Elemento
1	8	3
2	8	2
3	6	0
4	6	1
5	12	5
6	8	1
7	7	1
8	8	0
9	4	0
10	9	3
11	6	0
12	6	0
13	8	1
14	7	2
15	10	3
16	8	1
17	6	1
18	4	0
19	7	0
20	4	0

Tabla 3.-Resultados métrica Promedio

- Resultados Métrica Lesk

Iteración	Número de clústeres	Número de clústeres con un Elemento
1	6	4
2	2	0
3	2	0
4	6	3
5	2	0
6	2	1
7	4	2
8	2	0
9	6	4
10	4	2
11	2	0
12	6	3
13	2	0
14	2	0
15	2	0
16	2	0
17	2	0
18	2	0
19	5	2
20	4	2

Tabla 4.-Resultados métrica Lesk

- Resultados Métrica Lin

Iteración	Número de clústeres	Número de clústeres con un Elemento
1	2	0
2	3	1
3	2	0
4	2	0
5	5	2
6	2	0
7	3	1
8	2	0
9	2	0
10	4	1
11	2	0
12	2	0
13	2	0
14	2	0
15	2	0
16	2	0
17	2	0
18	3	1
19	2	1
20	5	3

Tabla 5.- Resultados métrica Lin

- Resultados Métrica Path

Iteración	Número de clústeres	Número de clústeres con un Elemento
1	12	4
2	6	0
3	11	3
4	7	2
5	16	8
6	11	3
7	10	3
8	8	1
9	7	0
10	10	3
11	7	1
12	8	0
13	12	3
14	8	1
15	8	1
16	7	1
17	6	1
18	6	0
19	5	0
20	9	1

Tabla 6.-Resultados métrica Path

- Resultados Métrica Wu-Palmer

Iteración	Número de clústeres	Número de clústeres con un Elemento
1	6	0
2	9	3
3	9	2
4	4	0
5	7	1
6	7	3
7	5	0
8	5	0
9	5	0
10	10	3
11	7	2
12	5	0
13	7	1
14	7	0
15	9	2
16	8	3
17	7	0
18	10	5
19	5	0
20	6	1

Tabla 7.- Resultados métrica Wu-Palmer

En base a los resultados obtenidos se procede a realizar un análisis de Wilcoxon, este análisis es una prueba estadística para muestras no paramétricas, la cual nos indica si los resultados entre dos grupos de datos son diferentes o no [13]. El grupo de datos que comparamos son el número de cluster de cada métrica contra las demás, asi aplicando el método Wilcoxon se basa en dos hipotesis:

- HO: si los datos obtenidos son iguales, es decir no hay ningún cambio entre las métricas de similitud semántica

- Ha: son diferentes lo que significa que los datos de cada métrica son diferentes, por lo que podemos deducir que existe una métrica con la cual obtendremos mejores resultados.

Los resultados muestran que los análisis de Wilcoxon realizados al número de cluster obtenidos, demuestran que se rechaza la hipótesis inicial (HO) y se toma la hipótesis alterna (Ha). Se acepta la hipótesis alterna ya que las $W_{calculadas}$ es menor que las $W_{criticas,}$ como se puede observar en la Tabla 8, por lo tanto todas las muestras realizadas son diferentes, ninguna métricas es igual a la otra, o tiene los mismo resultados, con este resultado podemos afirmar que una de las matrices de similitud semántica es mejor que las demás.

Grupos de datos		n	P	$W_{calculada}$	$W_{cirtica}$
PromSimilarity	LeskSimilarity	17	P<0.001	2.5	34
PromSimilarity	LinSimilarity	20	P < 0.001	1.5	166
PromSimilarity	WuPalmerSimilarity	18	P > 0.2	70.5	123
PromSimilarity	PathSimilarity	18	0.005 < P < 0.01	27	130
LeskSimilarity	LinSimilarity	10	0.10 < P < 0.20	13	69
LeskSimilarity	WuPalmerSimilarity	18	P<0.001	8.5	123
LeskSimilarity	PathSimilarity	19	P < 0.001	0	152
LinSimilarity	WuPalmerSimilarity	20	P < 0.001	0	166
LinSimilarity	PathSimilarity	20	P < 0.001	0	166
WuPalmerSimilarity	PathSimilarity	17	0.02 < P < 0.05	29	104

Tabla 8.- Resultados prueba Wilcoxon

9. Conclusiones

En base a los resultados obtenidos en el análisis de Wilcoxon (Tabla 8), podemos afirmar que los datos son diferentes, es decir, no hay similitud entre las métricas por lo tanto solo existe una métrica que es mejor a las demás. La mejor matriz de similitud semántica para realizar la clasificación de los servicios web semánticos es la Promsimilarity como se puede observar en la Tabla 9, esta matriz se obtiene a través del promedio de las cuatro métricas, ya que obtuvo la mayor cantidad de clasificaciones correctas, así como se demostró que esta métrica es totalmente diferente a las demás.

Las metaheurísticas son una herramienta poderosa y que ofrece gran cantidad de aplicaciones, en este proyecto utilizamos la heurística de la colonia de hormigas, modificamos su estructura para plasmarla en una herramienta con la cual se desea mejorar la clasificación de servicios web semánticos, los resultados son prometedores para la continuación de este trabajo, creo que se dio un buen avance y se cubrieron las expectativas que se tenía sobre los algoritmos bio-inspirados.

El algoritmo ACO tiene muchas aplicaciones en la computación, en este trabajo iniciamos una nueva aplicación, que pretende mejorar e innovar la clasificación de servicios web semánticos, y los resultados arrojados en este trabajo, motivan un inicio prometedor para poder realizar.

Matriz de similitud	Número de ejecuciones	Número de ejecuciones con resultado correcto (Seis Clúster)
PromSimilarity	20	5
LinSimilarity	20	0
LeskSimilarity	20	4
WuPalmerSimilarity	20	2
PahtSimilarity	20	3

Tabla 9.-Resultados Finales

10. Anexo

- ## Anexo A: Pseudocódigo ACO modificado
  - ### Función Ejecute

```
%Variation of ant colony optimization for clustering
%program developes by Dr. Roman A. Mora

PROCESO ejecute
        res ← [ ]
        PARA  i ← 1 HASTA 20 REALIZA
           t_max ← 100 %máximo número de iteaciones
           n_ant ← 20   %numero de hormigas
           q_0 ← .95     %parametro de balance
           alpha ← 2
           beta ← 2
           ro ← 0.3
           distance_max ← 1
           M  ←  LEE Matriz de Datos
           solve_i ← [ ]
           solve_i ← ant_cluster(t_max,n_ant,q_0,alpha,beta,ro,distance_max,M)
           res ← [res solve_i]
        FIN PARA
        ESCRIBE res
FIN PROCESO
```

  - ### Función ant_cluster

```
PROCESO ant_cluster(t_max,n_ant,q_0,alpha,beta,ro,distance_max,M)

        r ← Tamaño (M)
        a_2 ← round(r/2)
        cluster ← [2:a_2]
        r2 ← Tamaño(cluster)
        tao_cluster ← r2/2  *  Unos[r2]
        tao_centroides ← 1/r  *  Unos[r]
```

vis_centriodes ← $\frac{1}{1 - \text{Promedio(M)}}$

vis_cluster ← $\left(\frac{\left(\text{cluster} - \text{Promedio(cluster)}\right)^2 + \frac{1}{r2}}{r2}\right)^{-1}$

mean_distance_i ← distance_max * Unos[r]

ds_distance ← $\sqrt{[\left(\frac{1}{12}\right) * distance\_max] * Unos[r]}$

S ← [ ]
S2 ← [ ]
PARA t ← 1 HASTA t_max REALIZA

a_cluster ← ∑[(tao_cluster)^{alpha} * (vis_cluster)^{beta}]
a_centroides ← ∑[(tao_centroides)^{alpha} * (vi_centroides)^{beta}]

p_cluster ← $\frac{\left((\text{tao\_cluster})^{alpha}\right) * \left((\text{vis\_cluster})^{beta}\right)}{\text{a\_cluster}}$

p_centroides ← $\frac{\left((\text{tao\_centroides})^{alpha}\right) * \left((\text{vis\_centroides})^{beta}\right)}{\text{a\_centroides}}$

solution ← Ceros[n_ant,r]
n_cluster ← [ ]
M_centroides ← Ceros[n_ant,r]
M_centroides1 ← Ceros[n_ant,r]
M_dis ← Ceros[n_ant,r]
f_costo ← [ ]
c1 ← [ ]
c2 ← [ ]
c3 ← [ ]
c4 ← [ ]
c5 ← [ ]
c6 ← [ ]
c7 ← [ ]
S1 ← [ ]
PARA i ← 1 HASTA n_ant REALIZA
visitados ← Ceros[r]
sol_i ← [ ]
SI Aleatorio ≤ q_0 % Aleatorio Numero entre 0.0 y 1.0
seletion_cluster ← Aleatorio
a1_cluster ← 1
p1_cluster ← p_cluster[1,a1_cluster]
MIENTRAS seletion_cluster > p1_cluster Y a1_cluster<r2
a1_cluster ← a1_cluster + 1
p1_cluster ← p1_cluster + p_cluster[1,a1_cluster]

```
   FIN MIENTRAS
   n_cluster[1,i] ← cluster[1,a1_cluster]
SINO
   n_cluster[1,i] ← round(2 + Aleatorio  *  (a_2-2))
FIN SI
PARA  ii ← 1    HASTA    n_cluster[1,i]  REALIZA
   SI Aleatorio < q_0
      selection_centriode ← Aleatorio
      a1_centroides ← 1
      p1_centroides ← p_centroides[1,a1_centroides]
       MIENTRAS  selection_centriode > p1_centroides Y
      a1_centroides < r  REALIZA
            a1_centroides ← a1_centroides + 1
           p1_centroides ← p1_centroides +
          p_centroides[1,a1_centroides]
      FIN MIENTRAS
      SI visitados[1,a1_centroides] == 0
         visitados[1,a1_centroides] ← 1
         M_centroides[i,a1_centroides] ← ii
         M_centroides1[i,a1_centroides] ← 1
         solution[i,a1_centroides] ← ii
      SINO
         contador ← 1
         MIENTRAS visitados[1,a1_centroides] == 1
         Y contador < r REALIZA
            selection_centriode ← Aleatorio
            a1_centroides ← 1
            p1_centroides ← p_centroides[1,a1_centroides]
            MIENTRAS  selection_centriode > p1_centroides
                  Y a1_centroides < r REALIZA
                  a1_centroides ← a1_centroides + 1
                  p1_centroides ← p1_centroides +
                  p_centroides[1,a1_centroides]
            FIN MIENTRAS
            contador ← contador + 1
            SI visitados[1,a1_centroides] == 0
               contador ← 0
            FIN SI
         FIN MIENTRAS
         SI contador == 0
            visitados[1,a1_centroides] ← 1
            M_centroides[i,a1_centroides] ← ii
            M_centroides1[i,a1_centroides] ← 1
            solution[i,a1_centroides] ← ii
         SINO
```

```
            a1_centroides ← round(1 + Aleatorio  *  (r-1)) %round
            redondea al entero más cercano
            MIENTRAS visitados[1,a1_centroides] == 1 REALIZA
               a1_centroides ← round(1 + Aleatorio  *  (r-1))
            FIN MIENTRAS
            visitados[1,a1_centroides] ← 1
            M_centroides[i,a1_centroides] ← ii
            M_centroides1[i,a1_centroides] ← 1
            solution[i,a1_centroides] ← ii
         FIN SI
      FIN SI
   SINO
      a1_centroides ← round(1 + Aleatorio  *  (r-1))
      MIENTRAS visitados[1,a1_centroides] == 1 REALIZA
         a1_centroides ← round(1 + Aleatorio  *  (r-1))
      FIN MIENTRAS
      visitados[1,a1_centroides] ← 1
      M_centroides[i,a1_centroides] ← ii
      M_centroides1[i,a1_centroides] ← 1
      solution[i,a1_centroides] ← ii

   FIN SI
   SI Aleatorio < q_0
      Aleatorio1 ← Aleatorio
      MIENTRAS Aleatorio1 == 0 REALIZA
         Aleatorio1 ← Aleatorio
      FIN MIENTRAS
      b ← √(-2  *  log (Aleatorio1)  *  cos(2  *  π  *  Aleatorio))
      a1_sim ← mean_distance_i[1,a1_centroides]
          + b  *  ds_distance[1,a1_centroides]
   SINO
      a1_sim ← Aleatorio  *  distance_max
   FIN SI
   SI a1_sim > distance_max
      a1_sim ← Aleatorio  *  distance_max
   FIN SI
   SI a1_sim < 0
      a1_sim ← Aleatorio  *  distance_max
   FIN SI
   M_dis[i,a1_centroides] ← a1_sim
FIN PARA
PARA  j ← 1    HASTA    r  REALIZA
   SI M_centroides[i,j] == 1
      PARA  jj ← 1    HASTA    r  REALIZA
```

```
            SI M[j,jj] < M_dis[i,j]
               SI visitados[1,jj] ≅ 1
                  visitados[1,jj] ← 1
                  solution[i,jj] ← solution[i,j]
               FIN SI
            FIN SI
         FIN PARA
      FIN SI
   FIN PARA
   sumvist ← ∑visitados
   SI sumvist/r ≅ 1
      PARA  j ← 1    HASTA    r  REALIZA
         SI M_centroides[i,j] ≅ 0
            PARA  jj ← 1    HASTA    r  REALIZA
               SI M[j,jj]< ← distance_max
                  SI visitados[1,jj] ≅ 1
                     visitados[1,jj] ← 1
                     solution[i,jj] ← solution[i,j]
                  FIN SI
               FIN SI
            FIN PARA
         FIN SI
      FIN PARA
   FIN SI
   sumvist ← ∑visitados
   SI sumvist/r ≅ 1
       PARA  j ← 1    HASTA    r  REALIZA
         SI visitados[1,j] ≅ 1
            a2_cluster ← round[1 + Aleatorio  *  [n_cluster[1,i]-1]]
            visitados[1,j] ← 1
            solution[i,j] ← a2_cluster
         FIN SI
      FIN PARA
   FIN SI
     (c1[i,1] c2[i,1] c3[i,1] c4[i,1] c5[i] c6[i] c7[i]) ← FUNCION e_costo
     (M,solution[i],distance_max,n_cluster[1,i],r,M_centroides[i])
FIN PARA
S ← [solution,M_centroides,n_cluster^Traspuesta,c1, c2, c3, c4]
S1 ← [n_cluster^Transpuesta,c1, c2, c3, c4]
fitness_ant ←  FUNCION costo (S1,n_ant)
[cbest_ant best_ant] ← MAXIMO[fitness_ant]
S2[t] ← S[best_ant]
PARA  j ← 1    HASTA    r REALIZA
```

```
            a_mean_distance ← ( MINIMO(c6[best_ant])
            MAXIMO(c7[best_ant]) Promedio(c5[ r]) Promedio(c5[best_ant])
            MINIMO(c6[best_ant]) MAXIMO[c7[best_ant]))
          mean_distance_iact[1,j] ← Promedio(a_Promedio_distance)
          ds_distance_act[1,j] ← √VARIANZA(a_Promedio_distance)
          SI mean_distance_iact[1,j] == NaN
             mean_distance_iact[1,j] ← distance_max
             ds_distance_act[1,j] ← 1
          FIN SI
        FIN PARA
        mean_distance_i ← ro * mean_distance_iact + (1-ro) *
      mean_distance_i
        ds_distance ← ro  *  ds_distance_act + (1-ro)  * ds_distance
        delta_cluster ← [ ]
        delta_cluster ← Ceros[1,r2]
        delta_c1 ← S1[best_ant,1]-1
        delta_cluster[1,delta_c1] ← cbest_ant
        delta_centriodes ← [ ]
        delta_centriodes ← cbest_ant  *  M_centroides1[best_ant]
        tao_cluster ← (tao_cluster  *  [1-ro]) + (ro  *  delta_cluster)
        tao_centroides ← (tao_centroides  *  [1-ro]) + (ro  *  delta_centriodes)
      FIN PARA
       REGRESA s_i ← S2[t_max]
FIN  PROCESO
```

- Función e_cost

```
PROCESO e_cost(M,solution,distance_max,n_cluster,r,M_centroides)

      mean_i ← CEROS[1,r] %double,float, double, double
      min_i ← CEROS[1,r]
      max_i ← CEROS[1,r]
      dist_i ← CEROS[1,r]
      violaciones_cluster ← CEROS[1,n_cluster]
      diferencia_cluster ← CEROS[1,n_cluster]
      lis ← [ ]
      lit_1 ← [ ]
      clusterwhitmoreoneelement ← 0
      PARA ii ← 1 HASTA n_cluster REALIZA
        a_1 ← [ ]
        a_2 ← [ ]
        a_4 ← [ ]
        j ← 1
        violaciones ← 0
        diferencia ← 0
```

```
   PARA i ← 1 HASTA r REALIZA
      SI M_centroides[1,i] == ii
         PARA iii ← 1 HASTA r REALIZA
            SI solution[1,iii] == M_centroides[1,i]
               lit_1[ii,j] ← iii
               lis[ii,j] ← M[i,iii]
               SI M[i,iii] > distance_max
                  diferencia ← diferencia + |distance_max-M[1,iii]|
               SINO
                  diferencia ← diferencia+  |distance_max-M[1,iii] |
               FIN SI
               a_1[j,:] ← M[iii,:]
               j ← j+1
            FIN SI
         FIN PARA
         PARA j1 ← 1 HASTA j-1 REALIZA
            a_3 ← lit_1[ii,j1]
            a_2[:,j1] ← a_1[:,a_3]
         FIN PARA
         SI j > 2
        clusterwhitmoreoneelement ← clusterwhitmoreoneelement+1
         FIN SI
         a_4 ← max[a_2]
         PARA j1 ← 1 HASTA j-1 REALIZA
            SI a_4[1,j1] > distance_max
               violaciones ← violaciones+1
            SINO
               violaciones ← violaciones+0
            FIN SI
         FIN PARA
         mean_i[1,i] ← ∑ lis[ii,:] / j-1

         min_i[1,i] ← MINIMO(lis[ii,:])
         max_i[1,i] ← MAXIMO(lis[ii,:])
         dist_i[1,i] ← VARIANZA(VARIANZA(a_2))
         violaciones_cluster[1,ii] ← violaciones
         diferencia_cluster[1,ii] ← diferencia
      FIN SI
   FIN PARA
FIN PARA

REGRESA e ← ∑ violaciones_cluster
REGRESA f ← diferencia_cluster[1,ii]
```

REGRESA a ← n_cluster-clusterwhitmoreoneelement
REGRESA b ← ∑ dist_i
REGRESA c ← mean_i
REGRESA d1 ← min_i
REGRESA d2 ← max_i

FIN PROCESO

- Función costo

PROCESO costo(S1,n_ant)

peor_1←MAXIMO(S1[:,1])
peor_2←MAXIMO(S1[:,2])
peor_3←MAXIMO(S1[:,3])

a_10←($\frac{1}{S1[:,1]}$*n_ant)*S1[:,5]

PARA i←1 HASTA n_ant REALIZA
SI S1[i,4] == 0
a_7[i,1]← $\left|S1[i,1]\text{-peor\_1}\right|$
a_8[i,1]← $\left|S1[i,2]\text{-peor\_2}\right|$
a_9[i,1]← $\left|S1[i,3]\text{-peor\_3}\right|$
SINO
a_7[i,1]←0
a_8[i,1]←0
a_9[i,1]←0
FIN SI
FIN PARA
a_12← ∑ a_7
SI a_12 == 0
a_12←1
FIN SI
a_13←∑ a_8
SI a_13 == 0
a_13←1
FIN SI
a_14← ∑ a_9
SI a_14 == 0
a_14←1
FIN SI
REGRESA cost_i ← $\frac{a\_7}{a\_12} + \frac{a\_8}{a\_13} + \frac{a\_9}{a\_14}$ - S1[:,4] + a_10
FIN PROCESO

- Anexo B: Código fuente Módulo de Recuperación de Datos
  - Clase mainTester

```
import java.io.IOException;
import java.io.PrintWriter;

import
org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyStorageException;

import jsc.descriptive.*;


public class mainTester
{

   public static void main(String[] args)
   {
         //primer paso Cargar la ontologia
          String localDir = "
/home/saul/NetBeansProjects/PTHormiga/Ontologias";
          String ontologyFile =
"/home/saul/NetBeansProjects/PTHormiga//Ontologias/Prueba.owl";
         String ontologyIRI =
"http://www.semanticweb.org/ontologies/2012/8/templateWSDL10.owl
";
         //Carga los datos de las operaciones desde la ontoligÃa
         OntologyManagement manager = new OntologyManagement();
         Operation [] serviceOper =
manager.LoadOntology(localDir, ontologyFile, ontologyIRI);

         //Realiza comparaciones de similitud entre las
operaciones

         Metrics med = new Metrics();
         try
         {
               PrintWriter writer_lex = new
PrintWriter("Operaciones_Servicios.txt", "UTF-8");
                         PrintWriter writer_Wu = new
PrintWriter("WuPalmerSimilarity.txt", "UTF-8");
               PrintWriter writer_Lin = new
PrintWriter("LinSimilatity.txt", "UTF-8");
                         PrintWriter writer_Path = new
PrintWriter("PathSimilarity.txt", "UTF-8");
                         PrintWriter writer_Lesk = new
PrintWriter("LeskSimilarity.txt", "UTF-8");
                         PrintWriter writer_prom = new
PrintWriter("promSimilarity.txt", "UTF-8");
```

```
            int n = serviceOper.length;
            int arraySize = ((n*n)-n)/2;

            //Se crean los arreglos con los datos resultantes
de las similitudes
            double [][] JacquardSimilarityArray = new
double[n][n];
            SimilarityObject [][] WuPalmerSimilarityArray =
new SimilarityObject[n][n];
            SimilarityObject [][] LinSimilarityArray = new
SimilarityObject[n][n];
            SimilarityObject [][] PathSimilarityArray = new
SimilarityObject[n][n];
            SimilarityObject [][] LeskSimilarityArray = new
SimilarityObject[n][n];

            int cont = 0;

        for (int i = 0; i< serviceOper.length; i++){
            for (int j=0; j< serviceOper.length; j++)
            {



//
                    double[][] semSim =
med.semSimWuPalmer(serviceOper[i], serviceOper[j]);
                    double WuSim = med.matchAverage(semSim);
                    WuPalmerSimilarityArray[i][j] = new
SimilarityObject(serviceOper[i].getOperationName(),
serviceOper[j].getOperationName(), WuSim , "Undefined");

writer_Wu.print(med.matchAverage(semSim)+" ");
//
//
                    double[][] semSimL =
med.semSimLin(serviceOper[i], serviceOper[j]);
                    double LinSim = med.matchAverage(semSimL);
                    LinSimilarityArray[i][j] = new
SimilarityObject(serviceOper[i].getOperationName(),
serviceOper[j].getOperationName(), LinSim , "Undefined");
                    writer_Lin.print(med.matchAverage(semSimL)+"
");
//
//
                    double[][] semSimP =
med.semSimPath(serviceOper[i], serviceOper[j]);
                    double PathSim = med.matchAverage(semSimP);
                    PathSimilarityArray[i][j] = new
SimilarityObject(serviceOper[i].getOperationName(),
serviceOper[j].getOperationName(), PathSim , "Undefined");
```

```
                    writer_Path.print(med.matchAverage(semSimP)+"
");
//
                    double[][] semSimLe =
med.semSimLesk(serviceOper[i], serviceOper[j]);
                    double LeskSim = med.matchAverage(semSimLe);
                    LeskSimilarityArray[i][j] = new
SimilarityObject(serviceOper[i].getOperationName(),
serviceOper[j].getOperationName(), LeskSim , "Undefined");

   writer_Lesk.print(med.matchAverage(semSimLe)+" ");
                    double  promedio
=(WuSim+LinSim+PathSim+LeskSim)/4;
                    writer_prom.print(promedio+" ");
               }
                   writer_lex.println(i+1 +".-
"+serviceOper[i].getOperationName()+" ");
                   writer_Wu.println();
                   writer_Lin.println();
                   writer_Path.println();
                   writer_Lesk.println();
                   writer_prom.println();
                  }

         //Se cierra el archivo de bitÃ¡cora
          writer_lex.close();
                   writer_Wu.close();
          writer_Lin.close();
                   writer_Path.close();
                   writer_Lesk.close();
                   writer_prom.close();
               System.out.println("El tamaÃ±o de servicios es: "
+ n);
         }
         catch(IOException e)
         {
               e.printStackTrace();
         }

   }


   public static double [][] getSimArray(SimilarityObject [][]
array)
   {
         double [][] res = new
double[array.length][array.length];

         for(int i=0;i<820; i++)
                     for(int j=0;j<820;j++)
               res[i][j] = array[i][j].getSim();
```

```
        return res;

    }
    public static double average(SimilarityObject M[][]){
          double suma = 0;
        double promedio = 0;
        double aux=0;
        double aux1=0;
        double var=0;
            System.out.println(M.length);

        int j=0;
            for (int i = 0; i <M.length; i ++)
            {
            for(j=0; j <M.length; j++)
            {
            suma = suma + M[ i ][ j ].getSim();
            }
            }
promedio = suma / ( M.length *j );

            for (int i = 0; i <M.length; i ++)
            {
            for(j=0; j <M.length; j++)
            {
             aux1=M[i][j].getSim()-promedio;
           aux= aux+(Math.pow(aux1,2));
            }
            }
             var=aux/(M.length*M.length);
            return var;
        }



    public static void printSimArray(double [] array)
    {
        for(int i=0; i<array.length; i++)
             System.out.println(array[i]);
    }

}
```

- Clase OntologyManagement

```
import java.io.File;
import java.util.Iterator;
import java.util.Set;

import org.semanticweb.owlapi.apibinding.OWLManager;
```

```
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLDataFactory;
import org.semanticweb.owlapi.model.OWLDataProperty;
import org.semanticweb.owlapi.model.OWLIndividual;
import org.semanticweb.owlapi.model.OWLLiteral;
import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLObjectProperty;
import org.semanticweb.owlapi.model.OWLOntology;
import
org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyIRIMapper;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.util.AutoIRIMapper;


public class OntologyManagement
{
   private IRI ontoIri;

   public Operation [] LoadOntology(String localDir, String
ontologyFile, String ontoloIRI)
   {
         Operation [] operations = null;

       try
       {
         //Primero se carga la ontologia
         File dir = new File( localDir );

            OWLOntologyManager man =
OWLManager.createOWLOntologyManager();
            OWLOntologyIRIMapper autoIRIMapper = new
AutoIRIMapper(dir, false);
            man.addIRIMapper(autoIRIMapper);

            File ontoFile = new File(ontologyFile);
            ontoIri = IRI.create(ontoFile);

            OWLOntology ontology =
man.loadOntologyFromOntologyDocument(ontoIri);

            IRI ontologyIRI = IRI.create(ontoloIRI);

            System.out.println("Ontologï¿½a cargada en memoria: "
+ ontology);
            System.out.println("IRI de la ontologï¿½a: " +
ontology.toString());

            IRI OperationClass =
IRI.create(ontologyIRI.toString() + "#" + "Operation");
```

```
          IRI InputParameter =
IRI.create(ontologyIRI.toString() + "#" + "hasParameterInput");
          IRI OutputParameter =
IRI.create(ontologyIRI.toString() + "#" + "hasParameterOutput");
          IRI ParameterPart = IRI.create(ontologyIRI.toString()
+ "#" + "hasPart");

             OWLDataFactory df =
OWLManager.getOWLDataFactory();
             OWLClass operationClass =
df.getOWLClass(OperationClass);
             OWLObjectProperty hasInputParameter =
df.getOWLObjectProperty(InputParameter);
             OWLObjectProperty hasOutputParameter =
df.getOWLObjectProperty(OutputParameter);
             OWLObjectProperty hasParameterPart =
df.getOWLObjectProperty(ParameterPart);

             //Lectura de datos de las operaciones una por una
             Set<OWLIndividual> operationIndividuals =
operationClass.getIndividuals(ontology);

             operations = new
Operation[operationIndividuals.size()];
             int i = 0;

             for(OWLIndividual singleOperation :
operationIndividuals)
            {

                  String operation = "";
                  String inputPar = "";
                  String inputParPart = "";
                  String outputPar = "";
                  String outputParPart = "";

              System.out.println();
                  operation =
getIndividualNameString(singleOperation);

                Set<OWLIndividual> inputParams =
singleOperation.getObjectPropertyValues(hasInputParameter,
ontology);
                  for (OWLIndividual inputParam : inputParams)
                  {
                  inputPar =
getIndividualNameString(inputParam);

                     Set<OWLIndividual> inputParts =
inputParam.getObjectPropertyValues(hasParameterPart, ontology);
                      for (OWLIndividual paramPart :
inputParts){
```

```
                        inputParPart =
getParameterPartName(paramPart);}
                    }

                    Set<OWLIndividual> outputParams =
singleOperation.getObjectPropertyValues(hasOutputParameter,
ontology);
                    for (OWLIndividual outputParam :
outputParams)
                    {
                        outputPar =
getIndividualNameString(outputParam);

                        Set<OWLIndividual> outputParts =
outputParam.getObjectPropertyValues(hasParameterPart, ontology);
                         for (OWLIndividual paramPart :
outputParts){
                         outputParPart =
getParameterPartName(paramPart);}
                    }

                    //Se crea el objeto operation
                    Operation op = new Operation(operation,
inputPar, inputParPart, outputPar, outputParPart);
                    System.out.println("Creando objeto operacion
" + op.getOperationName());
                    System.out.println("Parametro entrada " +
op.getInputMessage());
                    System.out.println("Parte parametro entrada "
+ op.getInputMessagePart());
                    System.out.println("Parametro salida " +
op.getOutputMessage());
                    System.out.println("Parte parametro salida "
+ op.getOutputMessagePart());

                    //Se agrega al arreglo que se devuelve
                    operations[i] = op;
                    i++;
            }
             System.out.println("Total: " +
operationIndividuals.size());


       }
       catch (OWLOntologyCreationException e)
       {
           System.out.println("No se pudo cargar la ontologï¿½a:
" + e.getMessage());
       }

       return operations;
   }
```

```
   public static void showInstances(OWLOntology ontology)
    {
               Set<OWLClass> classes;
            Set<OWLObjectProperty> objectProperties;
            Set<OWLDataProperty> dataProperties;

            classes = ontology.getClassesInSignature();
            dataProperties =
ontology.getDataPropertiesInSignature();
            objectProperties =
ontology.getObjectPropertiesInSignature();

            for(Iterator<OWLClass> it=classes.iterator();
it.hasNext();)
            {
                OWLClass clase = (OWLClass)it.next();
                System.out.println( "Individuos de la Clase " +
clase.toStringID());

            }
            System.out.println();

            for(Iterator<OWLObjectProperty> it=
objectProperties.iterator(); it.hasNext();)
            {
                OWLObjectProperty op =
(OWLObjectProperty)it.next();
                System.out.println("Propiedad de objeto " +
op.toStringID());
            }
            System.out.println();

            for(Iterator<OWLDataProperty> it=
dataProperties.iterator(); it.hasNext();)
            {
                OWLDataProperty dp = (OWLDataProperty)it.next();
                System.out.println( "Propiedad de dato " +
dp.toStringID());
            }
            System.out.println();
    }


   public IRI generateIRI(String suffix)
    {
        return IRI.create(ontoIri.toString() + "#" + suffix);
    }

   String getIndividualNameString (OWLIndividual ind)
   {
        String res = "";
```

```
            String aux = ind.toString();
            int i = aux.indexOf("#");
            int j = aux.indexOf("$");
            res = aux.substring(i+1, j);
            return res;
      }

      String getParameterPartName (OWLIndividual ind)
      {
            String res = "";
            String aux = ind.toString();
            int i = aux.lastIndexOf("$");
            int j = aux.indexOf(">");
            res = aux.substring(i+1, j);
            return res;
      }

   }
```

- Clase Metrics

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import edu.cmu.lti.lexical_db.ILexicalDatabase;
import edu.cmu.lti.lexical_db.NictWordNet;
import edu.cmu.lti.ws4j.RelatednessCalculator;
import edu.cmu.lti.ws4j.impl.HirstStOnge;
import edu.cmu.lti.ws4j.impl.JiangConrath;
import edu.cmu.lti.ws4j.impl.LeacockChodorow;
import edu.cmu.lti.ws4j.impl.Lesk;
import edu.cmu.lti.ws4j.impl.Lin;
import edu.cmu.lti.ws4j.impl.Path;
import edu.cmu.lti.ws4j.impl.Resnik;
import edu.cmu.lti.ws4j.impl.WuPalmer;
import edu.cmu.lti.ws4j.util.WS4JConfiguration;

public class Metrics
{

      //Metricas de similitud semantica

      public double[][] semSimWuPalmer(Operation Op1, Operation
Op2)
      {
            double res [][] = null;

            ILexicalDatabase db = new NictWordNet();
         RelatednessCalculator rcs =  new WuPalmer(db);
```

```
        WS4JConfiguration.getInstance().setMFS(true);

        String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
        String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));

        res = rcs.getNormalizedSimilarityMatrix(operationNameOp1,
operationNameOp2);
           return res;
     }

     public double[][] semSimLin(Operation Op1, Operation Op2)
     {
          double res [][] = null;

           ILexicalDatabase db = new NictWordNet();
        RelatednessCalculator rcs =  new Lin(db);

        WS4JConfiguration.getInstance().setMFS(true);

        String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
        String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));

        res = rcs.getNormalizedSimilarityMatrix(operationNameOp1,
operationNameOp2);
           return res;
     }

     public double[][] semSimPath(Operation Op1, Operation Op2)
     {
          double res [][] = null;

           ILexicalDatabase db = new NictWordNet();
        RelatednessCalculator rcs =  new Path(db);

        WS4JConfiguration.getInstance().setMFS(true);

        String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
        String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));

        res = rcs.getNormalizedSimilarityMatrix(operationNameOp1,
operationNameOp2);

           return res;
     }
```

```
      public double[][] semSimLesk(Operation Op1, Operation Op2)
      {
            double res [][] = null;

            ILexicalDatabase db = new NictWordNet();
          RelatednessCalculator rcs =  new Lesk(db);

          WS4JConfiguration.getInstance().setMFS(true);

          String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
          String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));

          res = rcs.getNormalizedSimilarityMatrix(operationNameOp1,
operationNameOp2);
            return res;
      }
      public void printSimilarityMatrix(double[][] m)
      {

            for(int i = 0; i < m.length; i++)
               {
                  for(int j = 0; j < m[i].length; j++)
                  {
                     System.out.printf("%f ", m[i][j]);
                  }
                  System.out.println();
               }
      }
      // funcion que regresa el el promedio de los tokens de cada
operacion
      public double matchAverage(double[][] m)
      {
            double res = 0.0;
            double acumT = 0.0;

            for(int i = 0; i < m.length; i++)
               {
                    double acumR = 0.0;
                    double contR = 0.0;
                    double promR = 0.0;

                  for(int j = 0; j < m[i].length; j++)
                  {
                   if(m[i][j] != 0.0)
                   {
                         acumR = acumR + m[i][j];
                         contR++;
                   }
                  }
```

```
                if(acumR != 0.0)
                  promR = (double) acumR/contR;

                acumT = acumT + promR;
              }
             int sumTokens = m.length + m[0].length;
              res = (2*acumT)/sumTokens;

           return 1-res;
      }


      //Metricas semanticas de lexico

    public float lexicalSimOperationNames(Operation Op1, Operation
Op2)
    {
        float lexSimOpName = 0;
        //Extraccion de sustantivos del nombre de las operaciones
        String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
        String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
        lexSimOpName = lexicalSimilarity(operationNameOp1,
operationNameOp2);

        return lexSimOpName;
    }

    //Calculo de la similitud de inputs de dos operaciones
    float lexicalSimInputs(Operation Op1, Operation Op2)
    {
        float inputSim = 0;

        //Extraccion de unidades lexicas del nombre de los
parametros de entrada
        String[] inputTokensOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getInputMessage()));
        String[] inputTokensOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getInputMessage()));

        //Calculo de la similitud lexica entre estas dos
operaciones
        inputSim = lexicalSimilarity(inputTokensOp1,
inputTokensOp2);

        return inputSim;
    }

    //     Calculo de la similitud de outputs de dos operaciones
    float lexicalSimOutputs(Operation Op1, Operation Op2)
    {
```

```
        float outputSim = 0;

        //Extraccion de unidades lexicas del nombre de los
parametros de salida
        String[] outputTokensOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOutputMessage()));
        String[] outputTokensOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOutputMessage()));

        //Calculo de la similitud lexica entre estas dos
operaciones
        outputSim = lexicalSimilarity(outputTokensOp1,
outputTokensOp2);

        return outputSim;
    }



    //Extraccion de tokens de una palabra compuesta
    private String[] stringLexicalUnits(String cad)
    {
        String[] result;

        // Busca por casos como get_flight_price y los deja como
get_Flight_Price
        Pattern p = Pattern.compile("_\\p{Lower}");
        Matcher m = p.matcher(cad);
        StringBuffer sb = new StringBuffer();
        while (m.find())
        {
            m.appendReplacement(sb, m.group().toUpperCase());
        }
        m.appendTail(sb);

        // get_Flight_Price --> getFlightPrice
        cad = sb.toString().trim().replace("_", "");

        //Separa todas las palabras getFlightPrice -->
[get][Flight][Price]
        p = Pattern.compile("\\p{Lower}\\p{Lu}");
        m = p.matcher(cad);
        sb = new StringBuffer();
        while (m.find())
        {
            m.appendReplacement(sb, m.group().charAt(0) + " " +
m.group().substring(1));
        }
        m.appendTail(sb);
        result = sb.toString().trim().split(" ");

        return result;
```

```
    }

    //Discriminante de palabras relevantes
    private String[] lexicalDiscriminant(String[] cad)
    {
         int k = 0;
        for (int i = 0; i < cad.length; i++)
        {
            //Discriminantes
            int com1 = cad[i].compareToIgnoreCase("http");
            int com2 = cad[i].compareToIgnoreCase("for");
            int com3 = cad[i].compareToIgnoreCase("Return");
            int com4 = cad[i].compareToIgnoreCase("Result");
            int com5 = cad[i].compareToIgnoreCase("_");
            int com6 = cad[i].compareToIgnoreCase("");
            int com7 = cad[i].compareToIgnoreCase("soap");

            if ((com1 != 0) && (com2 != 0) && (com3 != 0) && (com4
!= 0) && (com5 != 0) && (com6 != 0) && (com7 != 0))
            {
                k++;
            }
        }

        String cadRet[] = new String[k];
        int j = 0;

        for (int i = 0; i < cad.length; i++)
        {
            //Discriminantes
            int com1 = cad[i].compareToIgnoreCase("http");
            int com2 = cad[i].compareToIgnoreCase("for");
            int com3 = cad[i].compareToIgnoreCase("Return");
            int com4 = cad[i].compareToIgnoreCase("Result");
            int com5 = cad[i].compareToIgnoreCase("_");
            int com6 = cad[i].compareToIgnoreCase("");
            int com7 = cad[i].compareToIgnoreCase("soap");

            if ((com1 != 0) && (com2 != 0) && (com3 != 0) && (com4
!= 0) && (com5 != 0) && (com6 != 0) && (com7 != 0))
            {
                cadRet[j] = cad[i];
                j++;
            }
        }

        return cadRet;
    }

    //     Calculo de la similitud lexica entre arreglos de cadenas
    private float lexicalSimilarity(String[] cad1, String[] cad2)
    {
```

```
        float lexSimilarity = 0;

        int numTokOp1 = cad1.length;
        int numTokOp2 = cad2.length;

        int intersec = 0;

        //Calculo de la operacin de interseccion
        for (int i = 0; i < numTokOp1; i++)
        {
            for (int j = 0; j < numTokOp2; j++)
            {
                if ((cad1[i].compareToIgnoreCase(cad2[j]) == 0))
                {
                    intersec++;
                }

            }
        }

        //Calculo de la similitud
        int suma = cad1.length + cad2.length;
        int union = suma - intersec;
        lexSimilarity = (float) intersec / union;

        return lexSimilarity;
    }



}
```

- Objeto Clase Operación

```
public class Operation
{
    private String operationName;
    private String inputMessage;
    private String outputMessage;
    private String inputMessagePart;
    private String outputMessagePart;

    public Operation(String operationName, String inputMessage,
String inputMessagePart, String outputMessage, String
outputMessagePart)
    {
        this.operationName = operationName;
        this.inputMessage = inputMessage;
        this.outputMessage = outputMessage;
```

```
        this.inputMessagePart = inputMessagePart;
        this.outputMessagePart = outputMessagePart;
    }

    public String getOperationName()
    {
        return operationName;
    }

    public String getInputMessage()
    {
        return inputMessage;
    }

    public String getOutputMessage()
    {
        return outputMessage;
    }

    public String getInputMessagePart()
    {
        return inputMessagePart;
    }

    public String getOutputMessagePart()
    {
        return outputMessagePart;
    }



    @Override
    public String toString()
    {
        String s = "";
        s = s + "[Operation: " + operationName;
        s = s + "   - Input: " + inputMessage;
        s = s + "   - Output: " + outputMessage + "]";
        return s;
    }

}
```

- Clase Symilarity Object

```
public class SimilarityObject
{

    String Operation1;
    String Operation2;
    Double sim;
    String Desition;
```

```
     public SimilarityObject(String Operation1, String Operation2,
Double sim, String Desition)
     {
          this.Operation1 = Operation1;
          this.Operation2 = Operation2;
          this.sim = sim;
          this.Desition = Desition;
     }

     public String getOperation1() {
          return Operation1;
     }
     public void setOperation1(String operation1) {
          Operation1 = operation1;
     }
     public String getOperation2() {
          return Operation2;
     }
     public void setOperation2(String operation2) {
          Operation2 = operation2;
     }
     public Double getSim() {
          return sim;
     }
     public void setSim(Double sim) {
          this.sim = sim;
     }
     public String getDesition() {
          return Desition;
     }
     public void setDesition(String desition) {
          Desition = desition;
     }




}
```

- Clase parameter

```
public class Parameter
{
    String type;
    String name;

    public Parameter(String type, String name)
    {
        this.type = type;
        this.name = name;
    }
```

```
    public void setName(String name)
    {
        this.name = name;
    }

    public void setType(String type)
    {
        this.type = type;
    }

    public String getType()
    {
        return type;
    }

    public String getName()
    {
        return name;
    }

    @Override
    public String toString()
    {
        String s = "[Parameter: "+ type + " " + name + "]";
        return s;
    }


}
```

- Anexo C
  - Función Ejecute.m

```
%Variation of ant colony optimization for clustering
%program developes by Dr. Roman A. Mora
res=[];
for i=1:20
    t_max=100; %maximun number of iterations
    n_ant=20;   %number of ant
    q_0=.95;     %paramater of balace. it should take a value of
0 to 1
    alpha=2;
    beta=2;
    ro=0.3;
```

```
    distance_max=1;
    M=importdata('matriz.txt');
    solve_i=[];

solve_i=feval('ant_cluster',t_max,n_ant,q_0,alpha,beta,ro,distan
ce_max,M);
    res=[res; solve_i];
end
csvwrite('ifg_1.txt',res);
```

- Funcion ant_cluster.m

```
function[s_i]=ant_cluster(t_max,n_ant,q_0,alpha,beta,ro,distance
_max,M )
r=length(M);
a_2=round(r/2);
cluster=2:a_2;
r2=length(cluster);
tao_cluster=r2/2.*ones(1,r2);
tao_centroides=1/r.*ones(1,r);
vis_centriodes=1./(1-mean(M));
vis_cluster=(((cluster-mean(cluster)).^2+1/r2)./r2).^-1;
%M1=1-M;
mean_distance_i=(distance_max)*ones(1,r);
ds_distance=sqrt((1/12)*(distance_max))*ones(1,r);
S=[];
S2=[];
for t=1:t_max

a_cluster=sum(((tao_cluster).^alpha).*((vis_cluster).^beta));

a_centroides=sum(((tao_centroides).^alpha).*((vis_centriodes).^b
eta));

p_cluster=(((tao_cluster).^alpha).*((vis_cluster).^beta))./a_clu
ster;

p_centroides=(((tao_centroides).^alpha).*((vis_centriodes).^beta
))./a_centroides;
    solution=zeros(n_ant,r);
    n_cluster=[];
    M_centroides=zeros(n_ant,r);
    M_centroides1=zeros(n_ant,r);
    M_dis=zeros(n_ant,r);
    f_costo=[];
    c1=[];
    c2=[];
    c3=[];
    c4=[];
    c5=[];
    c6=[];
```

```
    c7=[];
    S1=[];
    for i=1:n_ant
        visitados=zeros(1,r);
        sol_i=[];
        if rand<=q_0
            seletion_cluster=rand;
            a1_cluster=1;
            p1_cluster=p_cluster(1,a1_cluster);
            while (seletion_cluster>p1_cluster)&&(a1_cluster<r2)
                a1_cluster=a1_cluster+1;
                p1_cluster=p1_cluster+p_cluster(1,a1_cluster);
            end
            n_cluster(1,i)=cluster(1,a1_cluster);
        else
            n_cluster(1,i)=round(2+rand*(a_2-2));
        end
        for ii=1:n_cluster(1,i)
            if rand<q_0;
                selection_centriode=rand;
                a1_centroides=1;
                p1_centroides=p_centroides(1,a1_centroides);
                while
(selection_centriode>p1_centroides)&&(a1_centroides<r)
                    a1_centroides=a1_centroides+1;

p1_centroides=p1_centroides+p_centroides(1,a1_centroides);
                end
                if visitados(1,a1_centroides)==0
                    visitados(1,a1_centroides)=1;
                    M_centroides(i,a1_centroides)=ii;
                    M_centroides1(i,a1_centroides)=1;
                    solution(i,a1_centroides)=ii;
                else
                    contador=1;
                    while
visitados(1,a1_centroides)==1&&(contador<r)
                        selection_centriode=rand;
                        a1_centroides=1;

p1_centroides=p_centroides(1,a1_centroides);
                        while
(selection_centriode>p1_centroides)&&(a1_centroides<r)
                                a1_centroides=a1_centroides+1;

p1_centroides=p1_centroides+p_centroides(1,a1_centroides);
                        end
                        contador=contador+1;
                        if visitados(1,a1_centroides)==0
                            contador=0;
                        end
                    end
```

```
                    if contador==0
                        visitados(1,a1_centroides)=1;
                        M_centroides(i,a1_centroides)=ii;
                        M_centroides1(i,a1_centroides)=1;
                        solution(i,a1_centroides)=ii;
                    else
                        a1_centroides=round(1+rand*(r-1));
                        while visitados(1,a1_centroides)==1
                            a1_centroides=round(1+rand*(r-1));
                        end
                        visitados(1,a1_centroides)=1;
                        M_centroides(i,a1_centroides)=ii;
                        M_centroides1(i,a1_centroides)=1;
                        solution(i,a1_centroides)=ii;
                    end
                end
            else
                a1_centroides=round(1+rand*(r-1));
                while visitados(1,a1_centroides)==1
                    a1_centroides=round(1+rand*(r-1));
                end
                visitados(1,a1_centroides)=1;
                M_centroides(i,a1_centroides)=ii;
                M_centroides1(i,a1_centroides)=1;
                solution(i,a1_centroides)=ii;

            end
            if rand<q_0;
                rand1=rand;
                while rand1==0
                    rand1=rand;
                end
                b=sqrt(-2*log(rand1))*cos(2*pi*rand);

a1_sim=mean_distance_i(1,a1_centroides)+b*ds_distance(1,a1_centr
oides);
            else
                a1_sim=rand*distance_max;
            end
            if a1_sim>distance_max
                a1_sim=rand*distance_max;
            end
            if a1_sim<0
                a1_sim=rand*distance_max;
            end
            M_dis(i,a1_centroides)=a1_sim;
        end
        for j=1:r
            if M_centroides(i,j)==1
                for jj=1:r
                    if M(j,jj)<M_dis(i,j)
                        if visitados(1,jj)~=1;
```

```
                            visitados(1,jj)=1;
                            solution(i,jj)=solution(i,j);
                        end
                    end
                end
            end
        end
        sumvist=sum(visitados(1,:));
        if sumvist/r~=1
            for j=1:r
                if M_centroides(i,j)~=0
                    for jj=1:r
                        if M(j,jj)<=distance_max
                            if visitados(1,jj)~=1;
                                visitados(1,jj)=1;
                                solution(i,jj)=solution(i,j);
                            end
                        end
                    end
                end
            end
        end
        sumvist=sum(visitados(1,:));
        if sumvist/r~=1
             for j=1:r
                if visitados(1,j)~=1
                    a2_cluster=round(1+rand*(n_cluster(1,i)-1));
                    visitados(1,j)=1;
                    solution(i,j)=a2_cluster;
                end
            end
        end
        [c1(i,1) c2(i,1) c3(i,1) c4(i,1) c5(i,:) c6(i,:)
c7(i,:)]=feval('e_costo',M(:,:),solution(i,:),distance_max,n_clu
ster(1,i),r,M_centroides(i,:));
        %S(i,:)=[solution(i,:),n_cluster,c1(i,1) c2(i,1)
c3(i,1)];
    end
    S(:,:)=[solution,M_centroides,n_cluster',c1, c2, c3, c4];
    S1=[n_cluster',c1, c2, c3, c4];
    fitness_ant=feval('costo',S1,n_ant);
    [cbest_ant best_ant]=max(fitness_ant);
    S2(t,:)=S(best_ant,:);
    for j=1:r
        a_mean_distance=([ min(c6(best_ant,:))
max(c7(best_ant,:)) mean(c5(:,r)) mean(c5(best_ant,:))
min(c6(best_ant,:)) max(c7(best_ant,:))]);
        mean_distance_iact(1,j)=mean(a_mean_distance);
        ds_distance_act(1,j)=sqrt(var(a_mean_distance));
        if mean_distance_iact(1,j)==NaN
             mean_distance_iact(1,j)=distance_max;
             ds_distance_act(1,j)=1;
```

```
        end
    end
    mean_distance_i=ro.*mean_distance_iact+(1-
ro).*mean_distance_i;%mean_distance_iact;%ro.*mean_distance_iact
+(1-ro).*mean_distance_i;
    ds_distance=ro*ds_distance_act+(1-
ro).*ds_distance;%ds_distance_act;%ro*ds_distance_act+(1-
ro).*ds_distance;
    delta_cluster=[];
    delta_cluster=zeros(1,r2);
    delta_c1=S1(best_ant,1)-1;
    delta_cluster(1,delta_c1)=cbest_ant;
    delta_centriodes=[];
    delta_centriodes=cbest_ant.*M_centroides1(best_ant,:);
    tao_cluster=(tao_cluster.*(1-ro))+(ro.*delta_cluster);
    tao_centroides=(tao_centroides.*(1-
ro))+(ro.*delta_centriodes);
end
s_i=S2(t_max,:);
end
```

- Función e_costo.m

```
function [a b e f c d1
d2]=e_costo(M,solution,distance_max,n_cluster,r,M_centroides)
mean_i=zeros(1,r)
min_i=zeros(1,r);
max_i=zeros(1,r);
dist_i=zeros(1,r);
violaciones_cluster=zeros(1,n_cluster);
diferencia_cluster=zeros(1,n_cluster);
lis=[];
lit_1=[];
clusterwhitmoreoneelement=0;
for ii=1:n_cluster
    a_1=[];
    a_2=[];
    a_4=[];
    j=1;
    violaciones=0;
    diferencia=0;
    for i=1:r
        if M_centroides(1,i)==ii
            for iii=1:r
                if solution(1,iii)==M_centroides(1,i);
                    lit_1(ii,j)=iii;
                    lis(ii,j)=M(i,iii);
                    if M(i,iii)>distance_max;
```

```
                        diferencia=diferencia+abs(distance_max-
M(1,iii));
                    else
                        diferencia=diferencia+abs(distance_max-
M(1,iii));
                    end
                    a_1(j,:)=M(iii,:);
                    j=j+1;
                end
            end
            for j1=1:j-1
                a_3=lit_1(ii,j1);
                a_2(:,j1)=a_1(:,a_3);
            end
            if j>2

clusterwhitmoreoneelement=clusterwhitmoreoneelement+1;
            end
            a_4=max(a_2);
            for j1=1:j-1
                if a_4(1,j1)>distance_max
                    violaciones=violaciones+1;
                else
                    violaciones=violaciones+0;
                end
            end
            mean_i(1,i)=sum(lis(ii,:))/(j-1);
            min_i(1,i)=min(lis(ii,:));
            max_i(1,i)=max(lis(ii,:));
            dist_i(1,i)=var(var(a_2));
            violaciones_cluster(1,ii)=violaciones;
            diferencia_cluster(1,ii)=diferencia;
        end
    end

end
e=sum(violaciones_cluster);
f=diferencia_cluster(1,ii);
a=(n_cluster-clusterwhitmoreoneelement);
c=mean_i;
d1=min_i;
d2=max_i;
```

- Funcion costo.m

```
function [cost_i]=costo(S1,n_ant)
peor_1=max(S1(:,1));
peor_2=max(S1(:,2));
peor_3=max(S1(:,3));
a_10=(1./(S1(:,1).*n_ant)).*S1(:,5);
```

```
for i=1:n_ant
       if S1(i,4)==0
        a_7(i,1)=abs(S1(i,1)-peor_1);
        a_8(i,1)=abs(S1(i,2)-peor_2);
        a_9(i,1)=abs(S1(i,3)-peor_3);
       else
        a_7(i,1)=0;
        a_8(i,1)=0;
        a_9(i,1)=0;
       end
end
a_12=sum(a_7);
if a_12==0
    a_12=1;
end
a_13=sum(a_8);
if a_13==0
    a_13=1;
end
a_14=sum(a_9);
if a_14==0
    a_14=1;
end
cost_i=(a_7./a_12)+(a_8./a_13)+(a_9./a_14)-S1(:,4)+a_10;
%[a cost_i]=max(a_9);
```

11. Referencias bibliográficas

[1] C. B. Pop, V. R. Chifu, I. Salomine, M. Dinsoreanu, T. David, V. Acretoaie, A. Nagy y C. Oprisa. "Biologically inspired clustering of semantic Web services. Birds or ants intelligence?" Universidad Técnica de Cluj-Napoca, Romania, Octubre 2, 2011.

[2] Vittorio Maniezzo, Luca Maria Gambarde, Fabio de Luigi. "Ant Colony Optimization" [En Linea]. Disponible: http://www.cs.unibo.it/bison/publications/ACO.pdf

[3] Dorigo M., V. Maniezzo, A. Colorni ."Positive Feedback as a Search Strategy". Technical Report No. 91-016, Politecnico di Milano, Italy, 1991.
[4] Nicolas Nicolov, Kalina Bontcheva, Galia Angelova, Ruslan Mitkov. "Recent Advances in Natural Language Processing IV", John Benjamins Publishing, 2007 [En Linea]. Disponible: http://books.google.com.mx/books?id=P-1GqymUPIgC&dq=Recent+Advances+in+Natural+Language+Processing+IV&hl=es&source=gbs_navlinks_s

[5] E. Sánchez Estrada. "Clasificación de servicios web mediante ontologías", Propuesta de Proyecto Terminal, Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2012.

[6] J. Pascual Martínez. "Extracción automatizada y representación de servicios Web mediante ontologías", Proyecto Terminal, Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2011.

[7] M. García Sierra. "Sistema clasificador de ontologías mediante métodos de máquinas de soporte vectorial", propuesta de proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2013.

[8] J. A. Guzmán Luna, I. D. Torres Pardo, y J. C. Jaramillo Alzate. "Sistema de recuperación de servicios web basado en un módelo de navegación taxonómica.", Revista Colombiana de Tecnologías Avanzada, Colombia. 2010. [En línea]. Disponible: http://www.unipamplona.edu.co/unipamplona/portalIG/home_40/recursos/03_v13_18/revista_16/27102011/17.pdf

[9] R. A. Guzmán Ríos. "Arquitectura de servicios web semánticos sensible al contexto para dispositivos móviles", Tesis de maestría, Centro de Investigación de Estudios Avanzados del Instituto Politécnico Nacional, D.F., México, 2012. [En línea].Disponible: http://www.cs.cinvestav.mx/TesisGraduados/2012/TesisRafaelGuzman.pdf

[10] Yucheng Kao, Kevin Cheng. "An ACO-based clustering algorithm", ANTS'06 Proceedings of the 5th international conference on Ant Colony Optimization and Swarm Intelligence, Springer-Verlag Berlin, Heidelberg, Septiembre, 2006. [En Linea]. Disponible: http://link.springer.com/chapter/10.1007/11839088_31

[11] C. B. Pop, V. R. Chifu, I. Salomine, M. Dinsoreanu, T. David, V. Acretoaie, A. Nagy y C. Oprisa. "Ant-based methods for semantic web service discovery and composition" Universidad Técnica de Cluj-Napoca, Romania, Octubre, 2010. [En línea]. Disponible: http://www.ubicc.org/files/pdf/521_521.pdf

[12] Marco Dorigo. “Ant Colony Optimization”. Universidad Libre de Bruselas, Bruselas, Bélgica. Disponible: http://www.aco-metaheuristic.org/

[13] Devore L. Jay. “Probabilidad y Estadística para Ingenieria y Ciencias” Septima Edición, capitulo 8, Cengage Learning.