Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Enlazando datos abiertos académicos y profesionales a partir de recursos Web

Proyecto Tecnológico

Guillermo Fermín Poma Hernández

Matrícula: 2113031773

al2113031773@alumnos.azc.uam.mx

Asesor:

José Alejandro Reyes Ortiz

Profesor Titular

Departamento de Sistemas

jaro@correo.azc.uam.mx

Trimestre 16-O

Fechas de entrega:

9 de enero de 2017

Yo, José Alejandro Reyes Ortiz, 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 la UAM Azcapotzalco.

Yo, Guillermo Fermín Poma Hernández, 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.

Resumen

Hoy en día la búsqueda de información crece de forma exponencial, el interés de los usuarios por obtener nuevas fuentes de información es mayor.
Pero en la web existe ya tanta información de distintos orígenes, centros de investigación, blogs, buscadores, etc. Que cada vez se nos hace menos confiable y más engorroso ya buscar información.

Cuando se trata de buscar un tema en específico, navegamos por distintos buscadores y mediante estos en distintas fuentes de información también y esto ya hasta no puede desviar del objetivo de búsqueda inicial.

Esta herramienta se desarrolló con el fin de sólo tener una fuente de información que para este caso de usa “Google Académico” que está especializado en temas de autores y artículos científicos, y obtener nueva información con los datos que se van capturando mientras el usuario navega por la aplicación y así obtener una nueva forma de búsqueda, nueva relaciones sobre los temas que busca el usuario utilizando Ontologías.

Índice General

Índice de Figuras

Índices de Anexos

1. Introducción

Las personas tienen la necesidad de buscar información en la web sobre información académica. Esta tarea es tediosa y toma de mucho tiempo encontrar la información que buscan. Las causas de estos problemas, es que se navega por diferentes páginas web arrojadas por el buscador, muchas de estas no son de fuentes confiables y/o no son del tema que se desea.

Además, localizar, específicamente, información sobre publicaciones o autores involucra una ardua tarea debido a la navegación por varios sitios y con una gran diversidad de formatos. Lo que se necesita es contar con una estructura única que unifique los diversos sitios donde se encuentra la información sobre publicaciones, autores, temas de interés, entre otros.

Por lo anterior, para solucionar este problema, el presente proyecto plantea el uso de Ontologías como organizador de información y dar una nueva forma de búsqueda de información obteniendo nueva información entre las relaciones que se obtengan con la búsqueda en la web, para este caso se usa la conexión al banco de publicaciones científicas de Google llamado Google Scholar.

2. Antecedentes

2.1. Proyectos terminales

2.1.1. Poblador Automático de Ontologías [1].

El presente proyecto captura los datos obtenidos de lecturas en formatos PDF y TXT para así crear relaciones semánticas y construir ontologías con las mismas.

Este trabajo se parece a nuestra propuesta en el poblar ontologías a partir de datos obtenidos de fuentes distintas y difiere en el tipo de datos usado, ya que, esta propuesta utiliza datos de sitios web.

2.1.2. Sistema Web para la identificación automática de aspectos académicos y de experiencia profesional en expedientes curriculares [2].

Este proyecto se propone extraer la información académica y experiencia profesional de los curriculums vitae descritos en español utilizando técnicas de Procesamiento de Lenguaje Natural (PLN), específicamente reglas sintácticas y semánticas. La información extraída será validada por un usuario en una interfaz web para su inserción en una base de datos.

Este trabajo se parece a nuestra propuesta en la obtención de datos académicos a partir de fuentes de información y difiere en las fuentes de datos con la cual se va a trabajar además de no utilizar ontologías.

2.1.3. Representación semántica y extracción de información sobre publicaciones y expedientes curriculares [3].

Este proyecto lee los expedientes curricular en formato PDF de ellos se extrae solo la parte de publicaciones o artículos y sobre lo extraído utilizando reglas gramaticales se extrae aspectos importantes como el año, título y los autores y esos se almacenan en una ontología.

Este trabajo se parece a nuestra propuesta en la propuesta en la obtención de datos académicos y relacionarlas en una ontología pero se difieren en las fuentes de datos que utiliza y la forma de relacionar los datos.

2.2. Tesis

2.2.1. Ontologías de control de autoridades en el ámbito de los datos abiertos enlazados [4].

Con base en el enfoque de la función que lenguajes documentales se desarrollan en el entorno de datos abiertos vinculados, dos ontologías para la publicación de archivos de control de la autoridad se analizan brevemente: Mads / RDF y la ontología GND. Considerando la necesidad de contemplar la realidad de datos abiertos vinculados en una perspectiva más integradora y global, en el que no sólo los conjuntos de datos están interconectados, sino también los vocabularios controlados y modelos de descripción.

2.3. Artículos científicos

2.3.1. Datos abiertos enlazados *linked open* data en documentación científica [5].

Con el presente trabajo se pretende situar el concepto de datos abiertos enlazados, *Linked Open Data (LOD)* y su relevancia como elemento base en los procesos de preservación, recuperación e intercambio de información.

3. Justificación:

Para muchos usuarios de internet, mayormente para los usuarios de los motores de búsquedas, les es complicado y atareado navegar entre distintas fuentes que según contienen la información que buscamos.

Lo que se propone en este proyecto es una herramienta que tenga acceso directo a fuentes de datos, 100% confiables y mostrarlo en una sola interfaz, con esto el usuario no debe de salir de la ventana y/o consultar de otras fuentes. Además con los datos que busque y que se obtengan se creará una relación de los datos para poder generar nueva búsqueda y nueva información de las mismas.

4. Objetivo general:

Desarrollar una aplicación que permita descubrir relaciones entre datos abiertos de perfiles académicos y profesionales disponibles en la web mediante una ontología del dominio académico y reglas.

4.1. Objetivos específicos:

4.1.1. Diseñar e implementar la ontología de dominio académico y profesional para la representación de datos.

4.1.2. Diseñar e implementar un método de descubrimiento de relaciones entre los datos abiertos académicos y profesionales mediante reglas que analicen el contenido de las fuentes de datos.

4.1.3. Diseñar e implementar un módulo para la representación automática de la información extraída en la ontología.

4.1.4.

5. Marco Teórico:

En este proyecto se hizo uso de distintas tecnologías como son:

5.1.1. Ontologías

Las ontologías (OWL) está pensando para procesar el contenido de la información. Esta herramienta permite una mejor extracción de información proporcionando una relación semántica formal.

5.1.2. NetBeans

NetBeans es un IDE de desarrollo libre enfocado principal en el desarrollo de aplicaciones en el lenguaje Java.

6. Desarrollo del Proyecto

Para el desarrollo del proyecto se utilizó como lenguaje de programación Java por la fácil manipulación del framwork OWL (dedicado al uso de ontologías) y como IDE de desarrollo se utilizó NetBeans por las herramientas que brinda para la programación en Java.

Se utiliza un diseño de ontología con las relaciones que debe tener la información a extraer.

El proyecto consta de 3 módulos las cuales se muestran en la Figura 1 y se describen a continuación.

Figura 1.- Arquitectura de la aplicación

6.1. Módulo de búsqueda de información

En este módulo se muestra todas las fases de búsqueda que debe de recorrer el usuario para obtener una información acertada. Comienza con introducir el nombre del autor, artículo o publicación a buscar (no es necesario llenar ambos campos ya que lo delimita el tipo de botón a elegir o el tipo de búsqueda a realizar como se muestra en la Figura 2).

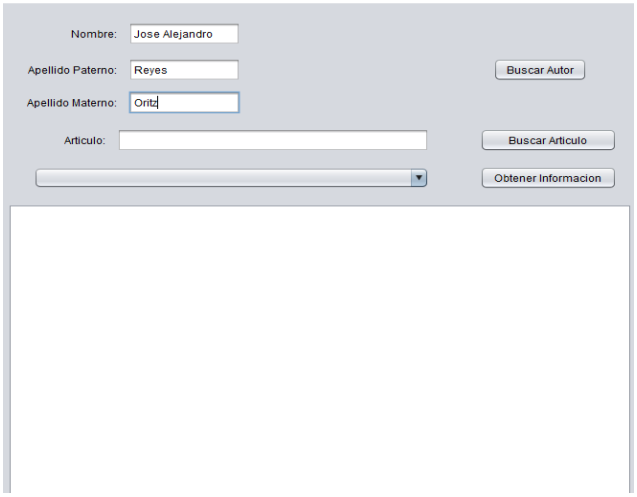


Figura 2. Imagen principal de la aplicación

Para continuar con la búsqueda, se colocó como ejemplo el nombre de mi aseso (José Alejandro Reyes Ortiz). A continuación se muestra el código en la Figura 3 con las etiquetas a buscar en el código html descargado desde la página de Google Scholar. El código completo de mostrará en el anexo A1.

```
//Liga que se concatena con el nombre del autor a buscar
private static final String GCR_SEARCH_THSIS_URL = "https://scholar.google.com/citations?view_op=search_authors&mauthors=";

//Etiqueta del bloque de los autores encontrados
private static final String THSIS_ELEMENT_CLASS = "gsc_1usr_text";

//Etiquetas con los valores que se desea obtener
private static final String THSIS_Nombre = "gsc_1usr_name";
private static final String THSIS_Titulo = "gsc_1usr_aff";
private static final String THSIS_Correo = "gsc_1usr_eml";
private static final String THSIS_Temas = "gsc_co_int";
private static final String THSIS_Url = "href";
```

Figura 3. Etiquetas a buscar autor el código html

Como se muestra en la figura anterior, se quiere obtener los siguientes valores: Nombre (nombre del autor), Título (título del autor o el nivel académico que muestra Google Scholar), Correo (no se obtiene el correo como tal, ya que Google Scholar no lo muestra, sólo muestra el dominio del correo), Temas (son las palabras relacionadas a los temas de investigación del autor) y Url (es la url de Google Sholar que nos permite acceder a las publicaciones del autor), todos estos valores son guardados en la clase “Autor” como se muestra en la Figura 4, código complementario se mostrará en el anexo A2.

```
public class Autor {

    private String nombre;
    private String titulo;
    private String correo;
    private String temas;
    private String url;
```

Figura 4. Clase “Autor”

Al hacer click en el botón “Buscar Autor” nos mostrará los resultados obtenidos por la consulta, que son los valores guardados en la clase Autor y estos valores son mostrados como lo indica la Figura 5.

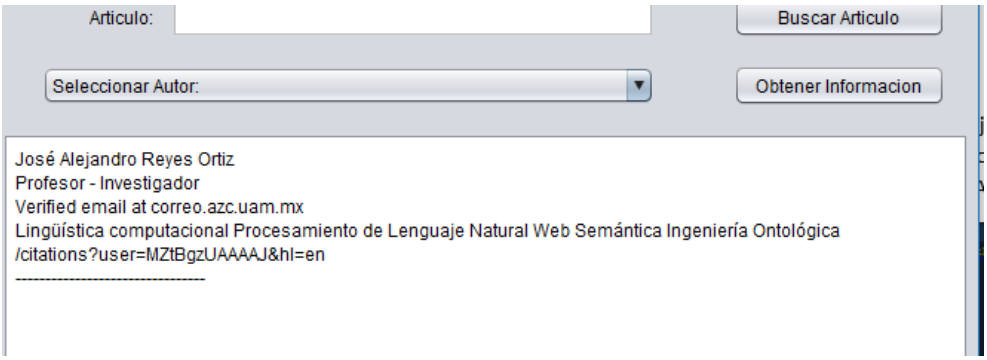


Figura 5. Datos obtenidos al buscar autor

Como comprobación, se hace la misma consulta en la página de Google Scholar y se muestra en la Figura 6.

Figura 6. Datos obtenidos desde Google Scholar

Para obtener más información del autor, como sus publicaciones, conferencias y colaboradores de del mismo se cuenta un combobox con los nombres de los autores que se encontraron con los nombres parecidos, para este caso sólo se obtuvo un valor como se muestra en la Figura 7.

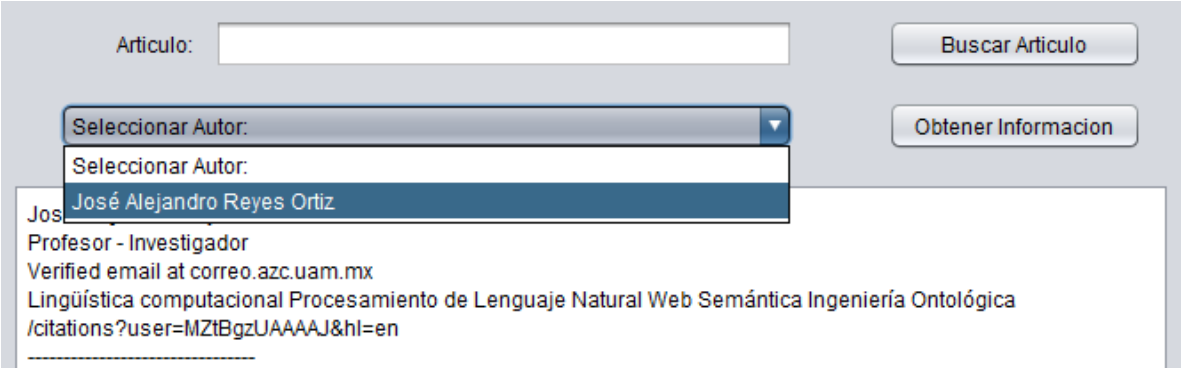


Figura 7. Lista de autores encontrados

Para obtener la información mencionada en el punto anterior, una vez seleccionado el autor de hace click en el botón “Obtener Información” la cual nos desplegará otra ventana con toda la información referida al autor seleccionado junto con sus publicaciones relacionadas como se muestra en la Figura 8.

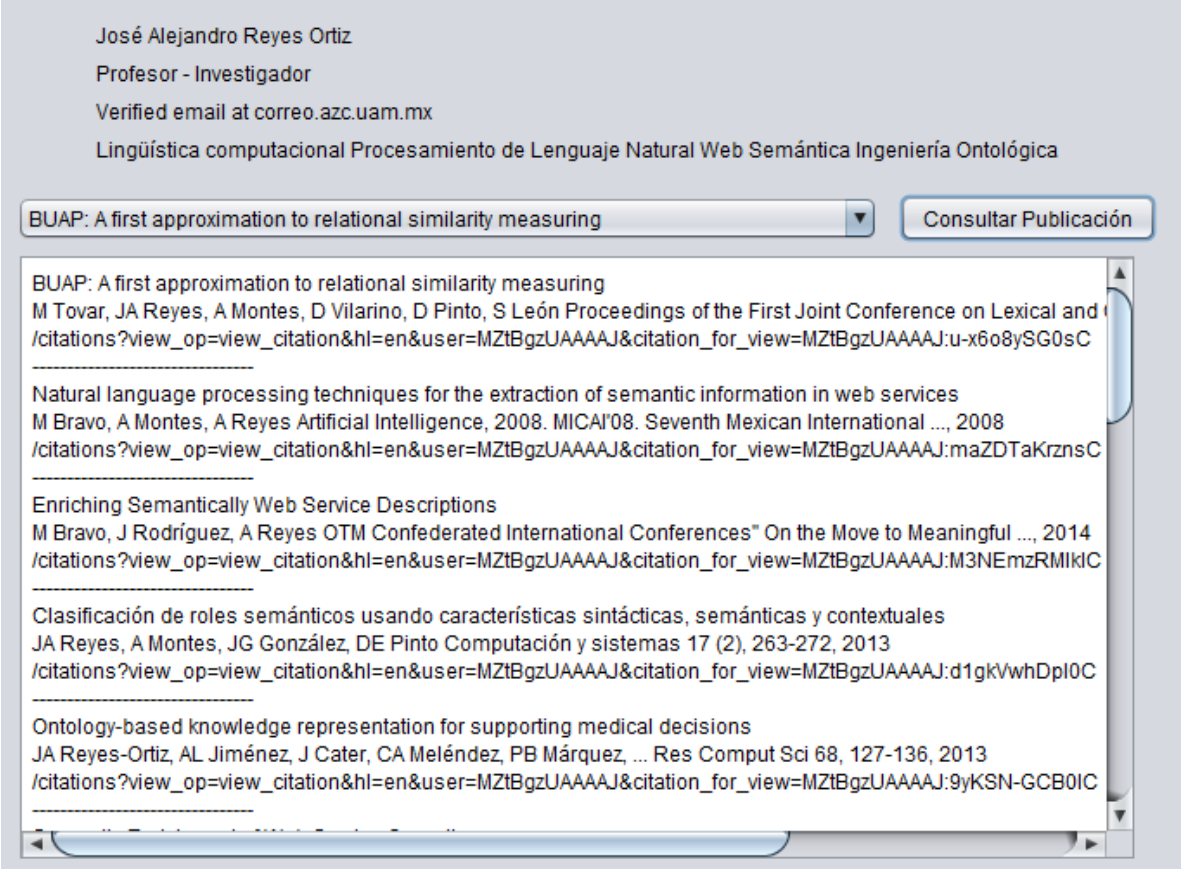
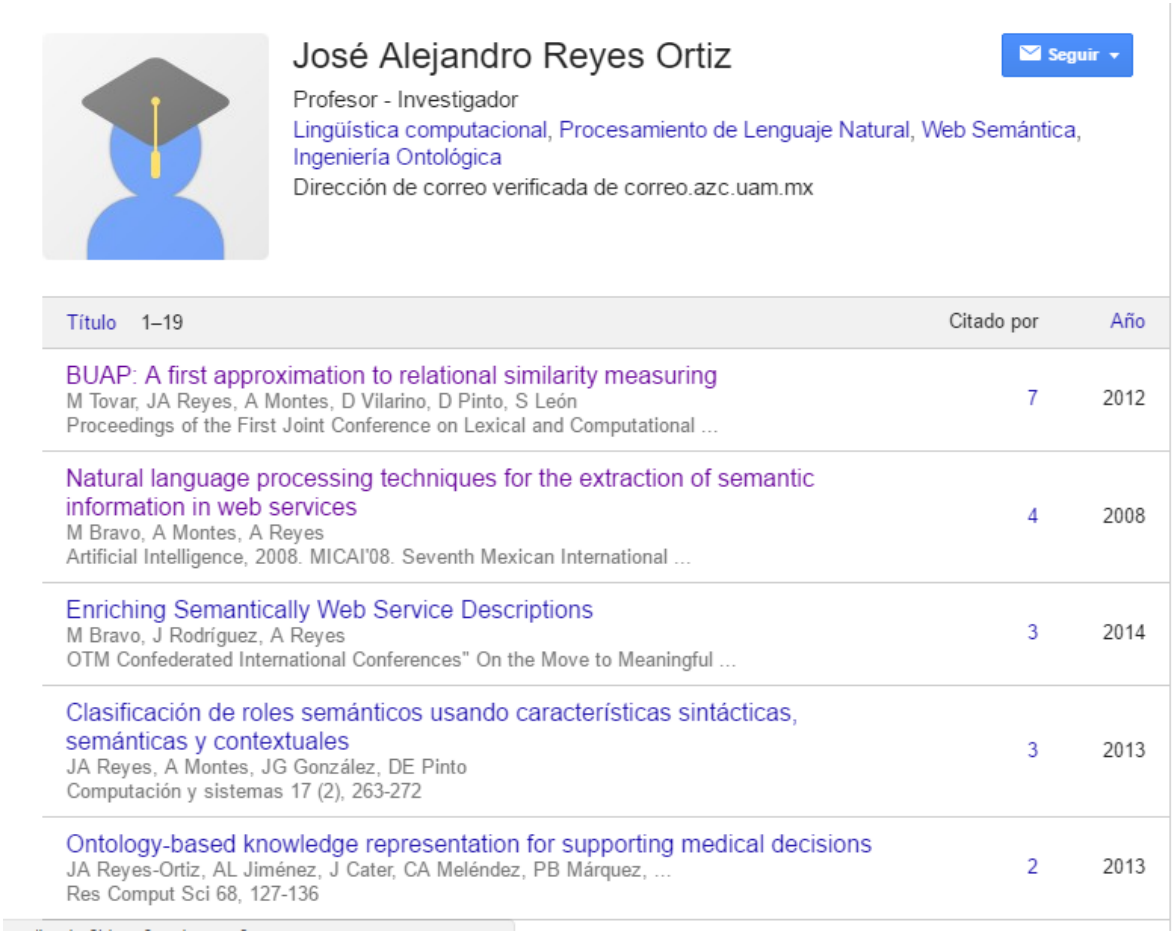


Figura 8. Obtención de información del autor

Como comprobación de la información obtenida, en la Figura 9 se muestra los resultados arrojados por las página de Google Scholar.



José Alejandro Reyes Ortiz

Seguir

Profesor - Investigador

Lingüística computacional, Procesamiento de Lenguaje Natural, Web Semántica, Ingeniería Ontológica

Dirección de correo verificada de correo.azc.uam.mx

Título 1–19	Citado por	Año
BUAP: A first approximation to relational similarity measuring M Tovar, JA Reyes, A Montes, D Vilarino, D Pinto, S León Proceedings of the First Joint Conference on Lexical and Computational ...	7	2012
Natural language processing techniques for the extraction of semantic information in web services M Bravo, A Montes, A Reyes Artificial Intelligence, 2008. MICAI'08. Seventh Mexican International ...	4	2008
Enriching Semantically Web Service Descriptions M Bravo, J Rodríguez, A Reyes OTM Confederated International Conferences" On the Move to Meaningful ...	3	2014
Clasificación de roles semánticos usando características sintácticas, semánticas y contextuales JA Reyes, A Montes, JG González, DE Pinto Computación y sistemas 17 (2), 263-272	3	2013
Ontology-based knowledge representation for supporting medical decisions JA Reyes-Ortiz, AL Jiménez, J Cater, CA Meléndez, PB Márquez, ... Res Comput Sci 68, 127-136	2	2013

Figura 9. Información obtenida desde Google Sholar

Para obtener esta información se realizó una lectura de etiquetas parecidas a la búsqueda del autor, estas etiquetas se muestran en la Figura 10, la descripción detallada de la manipulación de estas etiquetas se muestran más a detalle en el anexo A3.

```
//Liga que se concatena con el nombre del autor a buscar su informaciòm complementaria
private static final String GCR_SEARCH_THSIS_URL = "https://scholar.google.com";

//Lista de articulos encontrados
private static final String THSIS_ELEMENT_CLASS = "gsc_a_tr";

//Etiquetas con los valores que se desea obtener
private static final String THSIS_Titulo = "gsc_a_at";
private static final String THSIS_Colaboradores = "gs_gray";
```

Figura 10. Búsqueda de artículos relacionados y colaboradores

Como se muestra en la figura anterior, obtenemos los siguientes datos: Título (es el título de la publicación, colaboradores (son los colaboradores de la publicación). Estos datos son guardados en una clase “PublicacionAutor” como se muestra en la Figura 11, como observación al atributo contacto de obtiene dentro del código, el código completo se mostrará en el anexo A4.

```
public class PublicacionAutor {
    private String titulo;
    private String colaboradores;
    private String contacto;
    private String url;
```

Figura 11. Clase “PublicacionAutor”

Pasamos a obtener la información del artículo que se desea, seleccionando el artículo en el combobox y haciendo click en el botón consultar publicación como se muestra en la Figura 12, para este caso se selecciona el primer artículo mostrado, el código completo se mostrará en el Anexo A5.

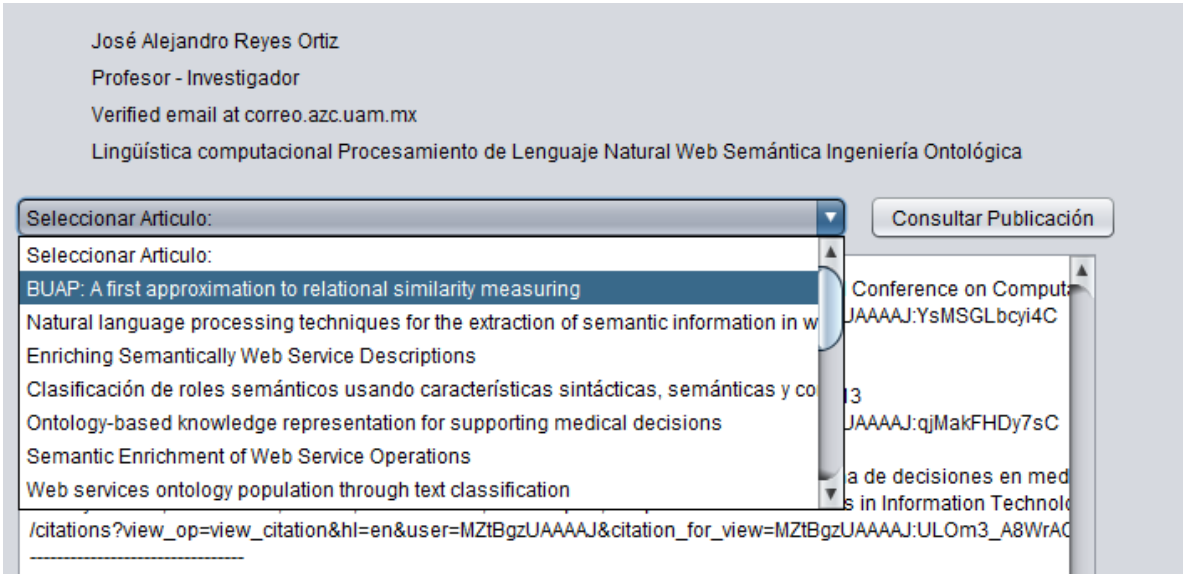


Figura 12. Selecciona artículo

Una vez realizado el paso anterior, nos aparece una nueva ventana con la información del artículo seleccionado como se muestra en la Figura 13.

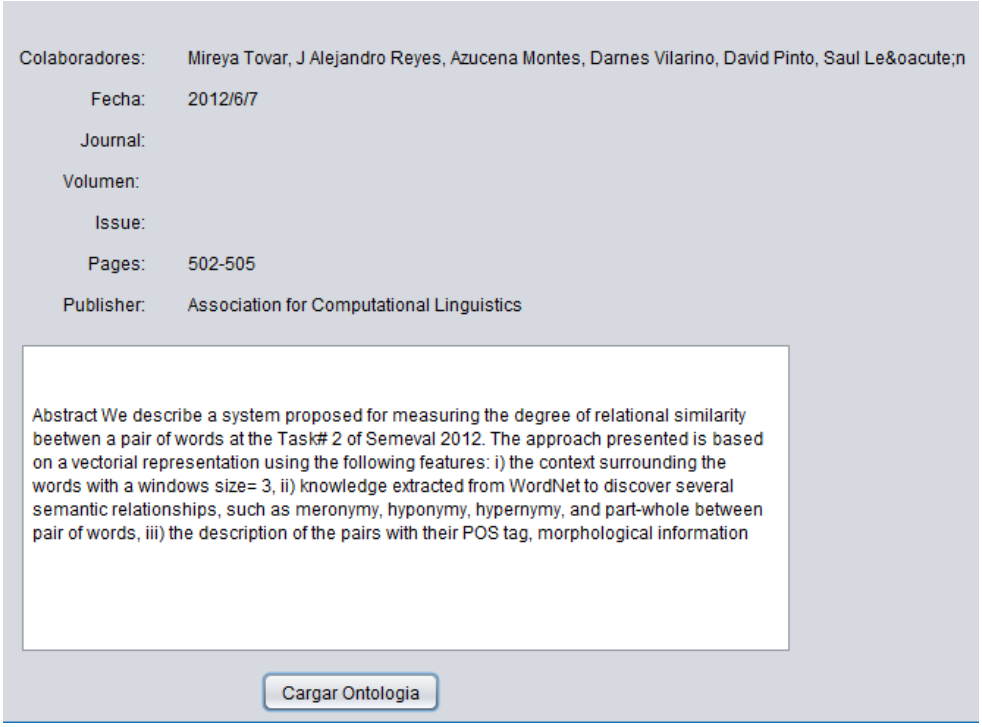


Figura 13. Información del artículo seleccionado

Como se puede observar no todos los datos están completos, como comprobación de los resultados obtenidos se hace la misma consulta en Google Scholar como se muestra en la Figura 14.

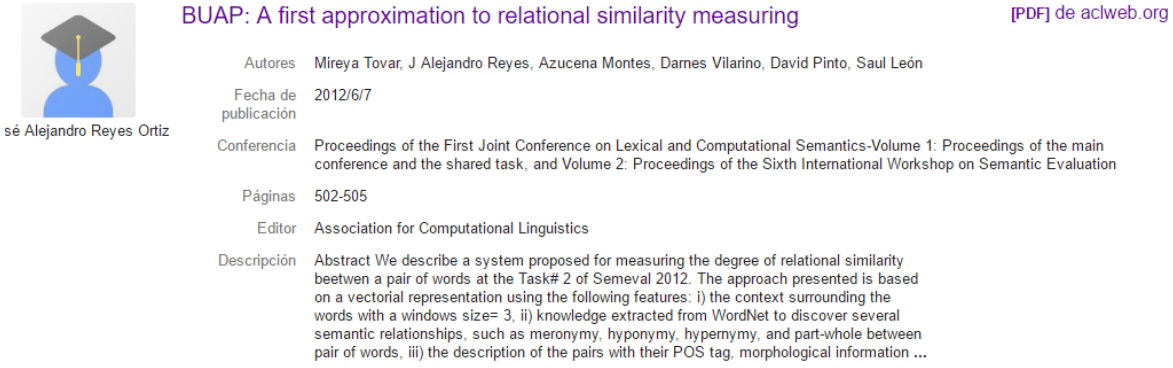


Figura 14. Información del artículo obtenida desde Google Scholar

Para obtener esta información sólo se usó una etiqueta que será mostrada en el anexo A6, pero para obtener los campos que se necesita se hace una comparación de la lista de información que se obtuvo como se muestra en la Figura 15.

```
for (int i = 0; i < thsisElements2.size(); i++){
    String e=thsisElements2.get(i).childNodes().get(0).toString();
if(e.equalsIgnoreCase("\nAuthors")){
    t.setColaboradores(thsisElements.get(i).childNode(0).toString());
}else if(e.equalsIgnoreCase("\nPublication date")){
    t.setFecha(thsisElements.get(i).childNode(0).toString());
}else if(e.equalsIgnoreCase("\nJournal")){
    t.setJournal(thsisElements.get(i).childNode(0).toString());
}else if(e.equalsIgnoreCase("\nVolume")){
    t.setVolumen(thsisElements.get(i).childNode(0).toString());
}else if(e.equalsIgnoreCase("\nIssue")){
    t.setIssue(thsisElements.get(i).childNode(0).toString());
}else if(e.equalsIgnoreCase("\nPages")){
    t.setPages(thsisElements.get(i).childNode(0).toString());
}else if(e.equalsIgnoreCase("\nPublisher")){
    t.setPublisher(thsisElements.get(i).childNode(0).toString());
}else if(e.equalsIgnoreCase("\nDescription")){
    int r = thsisElements.get(i).childNodeSize()-1;
    for (int j = 0; j < r; j=j+2){
        descripcion += "\n"+thsisElements.get(i).childNode(j).toString();
```

Figura 15. Obtener campos de la consulta del artículo

Toda esta información es guardada en la clase “ArticuloO” con los atributos que muestra la Figura 16, el código completo será mostrado en el anexo A7.

```
*/
public class ArticuloO {

    private String colaboradores;
    private String fecha;
    private String journal;
    private String volumen;
    private String issue;
    private String pages;
    private String publisher;
    private String description;
```

Figura 16. Clase “ArticuloO”

Como finalización de este módulo se hace click en el botón “Cargar Ontología”.

6.2. Módulo creación de relaciones

Como primera instancia se crean las clases ontológicas como se muestra en la Figura 17.

```
//Obtenemos las clases de la ontología
claseArticulo = factory.getOWLClass(IRI.create(url + "#Articulo"));
claseInstitucion = factory.getOWLClass(IRI.create(url + "#Institucion"));
claseInvestigador = factory.getOWLClass(IRI.create(url + "#Investigador"));
clasePublicacion = factory.getOWLClass(IRI.create(url + "#Publicacion"));
```

Figura 17. Creación de clases.

Como segundo paso se crean los ObjectProperty como se muestra en la Figura 18.

```
//Creamos los ObjectProperty
opEstaAdscritoA = factory.getOWLObjectProperty(IRI.create(ontologyIRI + "#estaAdscritoA"));
opFuePublicadoEn = factory.getOWLObjectProperty(IRI.create(ontologyIRI + "#fuePublicadoEn"));
opFuePublicadoPor = factory.getOWLObjectProperty(IRI.create(ontologyIRI + "#fuePublicadorPor"));
opColaboraCon = factory.getOWLObjectProperty(IRI.create(ontologyIRI + "#colaboraCon"));
```

Figura 18. Creación de ObjectProperty

El siguiente paso es crear los DataPRoperty como se muestra en la Figura 19.

```
//Obtenemos los DataProperty
dpTieneAnioDePublicacion = factory.getOWLDataProperty(IRI.create(url + "#tieneAnioDePublicacion"));
dpTieneAreaInvestigacion= factory.getOWLDataProperty(IRI.create(url + "#tieneAreaInvestigacion"));
dpTieneCorreo = factory.getOWLDataProperty(IRI.create(url + "#tieneCorreo"));
dpTieneNombreInstitucion = factory.getOWLDataProperty(IRI.create(url + "#tieneNombreInstitucion"));
dpTieneNombreInvestigador = factory.getOWLDataProperty(IRI.create(url + "#tieneNombreInvestigador"));
dpTieneNombrePublicacion = factory.getOWLDataProperty(IRI.create(url + "#tieneNombrePublicacion"));
dpTienePagInicial = factory.getOWLDataProperty(IRI.create(url + "#tienePagInicial"));
dpTienePosicionAcademica = factory.getOWLDataProperty(IRI.create(url + "#tienePosicionAcademica"));
dpTieneTitulo = factory.getOWLDataProperty(IRI.create(url + "#tieneTitulo"));
```

Figura 19. Creación de DataProperty

Una vez creadas las instancias, obtendremos los valores de la ontología creada anteriormente, la estructura de la ontología se explicará en el módulo Ontología, la lectura de la ontología se obtiene como se muestra en la Figura 20.

```
 private static final File direccion = new File("D:\\ProyrectoTerm\\OntologiaAcademica.owl");
 private static final String url ="http://www.semanticweb.org/alumno/ontologies/2016/0/OntologiaAcademica";

blic void leerOntologia() throws OWLOntologyStorageException, OWLOntologyCreationException {

 manejador = OWLManager.createOWLOntologyManager();
 ontologia = manejador.loadOntologyFromOntologyDocument(direccion);
```

Figura 20. Lectura de ontología ya creada.

Como se muestra en la figura anterior, obtenemos la ruta del archivo ".owl" y también la IRI que nos brinda el "Protege" al momento de crear la ontología.

A continuación se muestra la obtención de la IRI's de las clase de la ontología ya creada como se muestra en la Figura 21.

```
OWLIndividual owlAutor = factory.getOWLNamedIndividual(IRI.create(url + "#" + autor.getNombre().replaceAll(" ", "")));
OWLIndividual owlInstitucion = factory.getOWLNamedIndividual(IRI.create(url + "#" + autor.getTitulo().replaceAll(" ", "")));
OWLIndividual owlArticulo = factory.getOWLNamedIndividual(IRI.create(url + "#" + publicacion.getTitulo().replaceAll(" ", "")));
OWLIndividual owlPublicacion = factory.getOWLNamedIndividual(IRI.create(url + "#" + arti.getPublisher().replaceAll("\n", "").replaceAll(" ", "")));
```

Figura 21. Obtención de IRI's de las clases de la ontología

Ahora creaos la relación entre el DataProperty y la Clase con su respectivo valor como se muestra en la Figura 22.

```
OWLDataPropertyAssertionAxiom correo = factory.getOWLDataPropertyAssertionAxiom(dpTieneCorreo, owlAutor, autor.getCorreo().replaceAll("Verified email at "
OWLDataPropertyAssertionAxiom temas = factory.getOWLDataPropertyAssertionAxiom(dpTieneAreaInvestigacion, owlAutor, autor.getTemas());
OWLDataPropertyAssertionAxiom nombre = factory.getOWLDataPropertyAssertionAxiom(dpTieneNombreInvestigador, owlAutor, autor.getNombre());
OWLDataPropertyAssertionAxiom titulo = factory.getOWLDataPropertyAssertionAxiom(dpTienePosicionAcademica, owlAutor, autor.getTitulo());
OWLDataPropertyAssertionAxiom nomIStitu = factory.getOWLDataPropertyAssertionAxiom(dpTieneNombreInstitucion, owlInstitucion, autor.getTitulo());
OWLDataPropertyAssertionAxiom anio = factory.getOWLDataPropertyAssertionAxiom(dpTieneAnioDePublicacion, owlArticulo, arti.getFecha());
OWLDataPropertyAssertionAxiom pag = factory.getOWLDataPropertyAssertionAxiom(dpTienePagInicial, owlArticulo, arti.getPages());
OWLDataPropertyAssertionAxiom art = factory.getOWLDataPropertyAssertionAxiom(dpTieneTitulo, owlArticulo, publicacion.getTitulo());
OWLDataPropertyAssertionAxiom publi = factory.getOWLDataPropertyAssertionAxiom(dpTieneNombrePublicacion, owlPublicacion, arti.getPublisher().replaceAll("
```

Figura 22. Creación de relaciones Clases, DataProperty y valores.

Todo el código será explicado con mejor detalle en el Anexo B1.

Como siguiente paso se crean las relaciones de los ObjectProperty con las clases correspondientes como se muestra en la Figura 23.

```
OWLObjectPropertyAssertionAxiom AutorIntsti = factory.getOWLObjectPropertyAssertionAxiom(opEstaAdscritoA, owlAutor, owlInstitucion);
OWLObjectPropertyAssertionAxiom ArtiInves = factory.getOWLObjectPropertyAssertionAxiom(opFuePublicadoPor, owlArticulo, owlAutor);
OWLObjectPropertyAssertionAxiom Publica = factory.getOWLObjectPropertyAssertionAxiom(opFuePublicadoEn, owlArticulo, owlPublicacion);
```

Figura 23. Creacion de relaciones de los ObjectProperty con sus clases correspondientes.

Como último paso cargaremos todos los datos obtenidos en la búsqueda a la ontología como se muestra en la Figura 24.

```
AddAxiom addAxiomCorreo = new AddAxiom(ontologia, correo);
AddAxiom addAxiomTemas = new AddAxiom(ontologia, temas);
AddAxiom addAxiomNombre = new AddAxiom(ontologia, nombre);
AddAxiom addAxiomTitulo = new AddAxiom(ontologia, titulo);
AddAxiom addAxiomNomInsti = new AddAxiom(ontologia, nomIStitu);
AddAxiom addAxiomAutorInstitu = new AddAxiom(ontologia, AutorIntsti);
AddAxiom addAxiomAnio = new AddAxiom(ontologia, anio);
AddAxiom addAxiomPag = new AddAxiom(ontologia, pag);
AddAxiom addAxiomArt = new AddAxiom(ontologia, art);
AddAxiom addAxiomArtiInves = new AddAxiom(ontologia, ArtiInves);
AddAxiom addAxiomPubl = new AddAxiom(ontologia, publi);
AddAxiom addAxiomArtPubl = new AddAxiom(ontologia, Publica);
manejador.applyChange(new AddAxiom(ontologia, factory.getOWLClassAssertionAxiom(claseInvestigador, owlAutor)));
manejador.applyChange(new AddAxiom(ontologia, factory.getOWLClassAssertionAxiom(claseInstitucion, owlInstitucion)));
manejador.applyChange(new AddAxiom(ontologia, factory.getOWLClassAssertionAxiom(claseArticulo, owlArticulo)));
manejador.applyChange(new AddAxiom(ontologia, factory.getOWLClassAssertionAxiom(clasePublicacion, owlPublicacion)));
manejador.applyChange(addAxiomCorreo);
manejador.applyChange(addAxiomTemas);
manejador.applyChange(addAxiomNombre);
manejador.applyChange(addAxiomTitulo);
manejador.applyChange(addAxiomNomInsti);
manejador.applyChange(addAxiomAutorInstitu);
manejador.applyChange(addAxiomAnio);
manejador.applyChange(addAxiomPag);
manejador.applyChange(addAxiomArt);
manejador.applyChange(addAxiomArtiInves);
manejador.applyChange(addAxiomPubl);
manejador.applyChange(addAxiomArtPubl);
manejador.saveOntology(ontologia,manejador.getOntologyDocumentIRI(ontologia));
```

Figura 24. Cargando datos a ontología.

6.3. Módulo Ontología

Este módulo explica la creación de la ontología usada para este proyecto junto con sus respectivas relaciones, usando como ejemplo el llenado de datos lo procesado en el primer módulo con la búsqueda del asesor “José Alejandro Reyes Ortiz”.

6.3.1.Clase Artículo, esta clase muestra en la Figura 25 sus respectivas relaciones

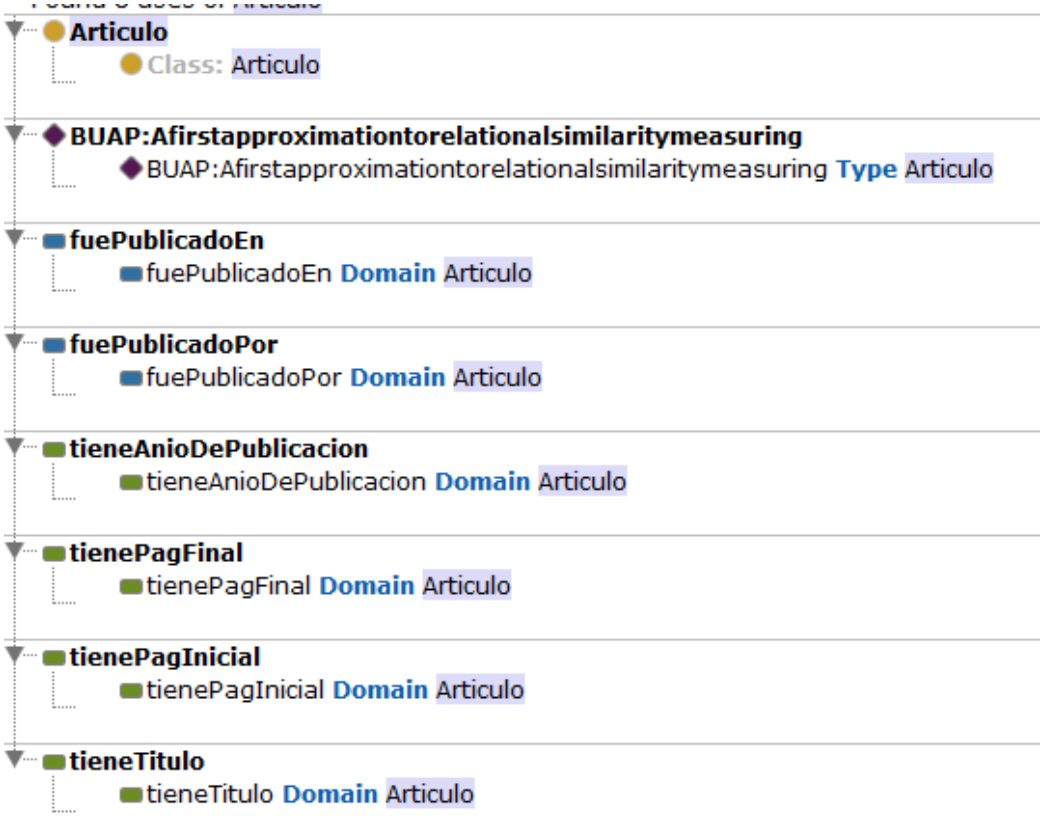


Figura 25. Propiedades de la clase Artículo

6.3.2.Clase Institución, las propiedades de esta clase se muestra en la Figura 26.

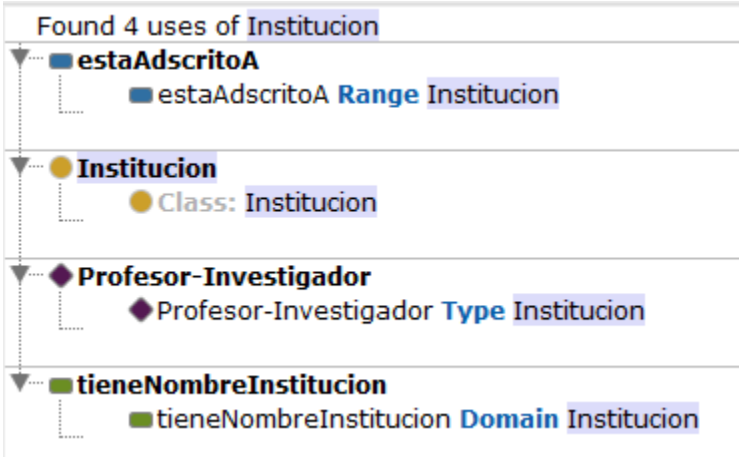


Figura 26. Propiedades de la clase Institucion

6.3.3.Clase Investigador, las propiedades de esta clase se muestra en la Figura 27.

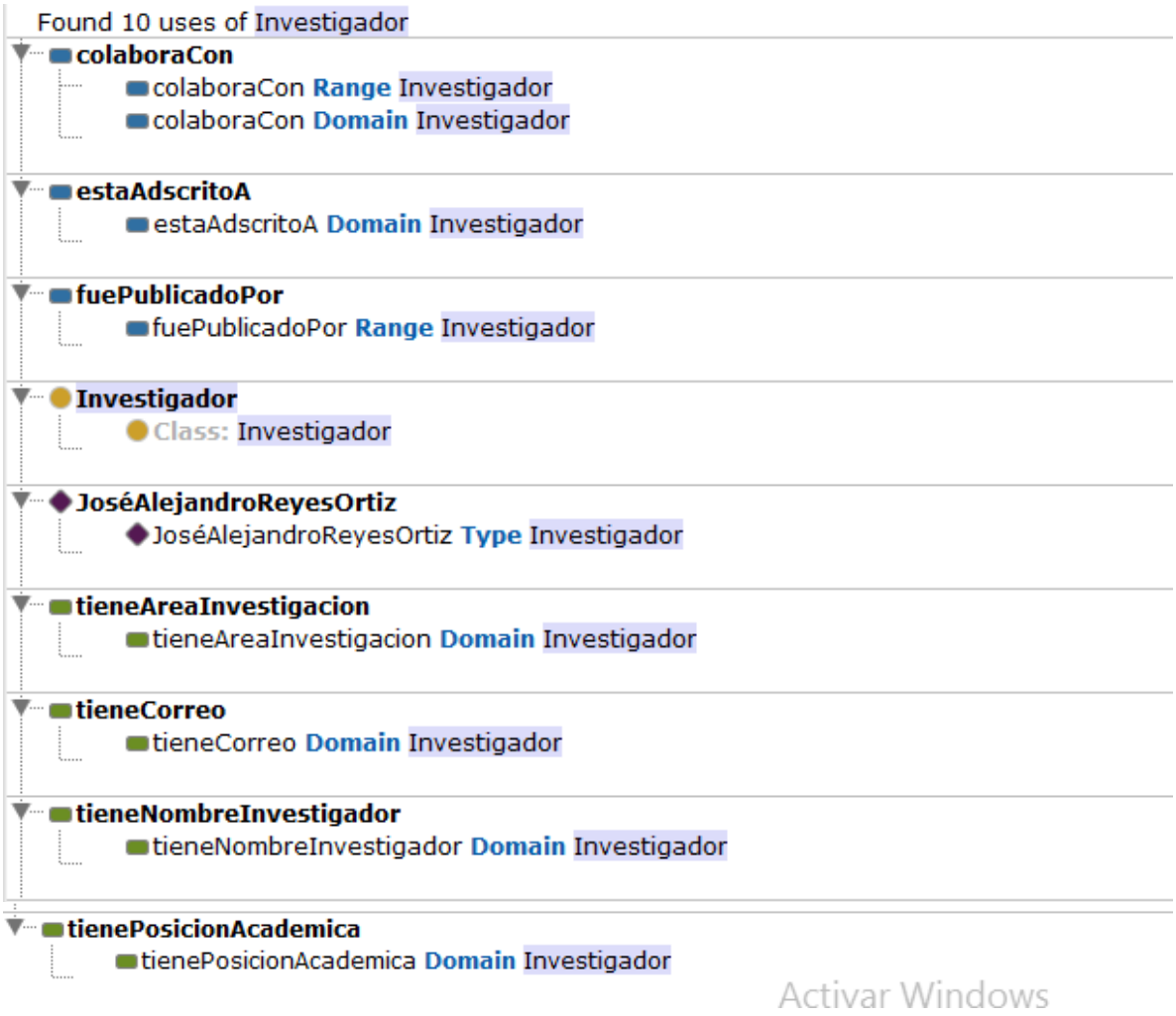


Figura 27. Propiedades de la clase Investigador

6.3.4.Clase Publicación, las propiedades de esta clase se muestra en la Figura 28.

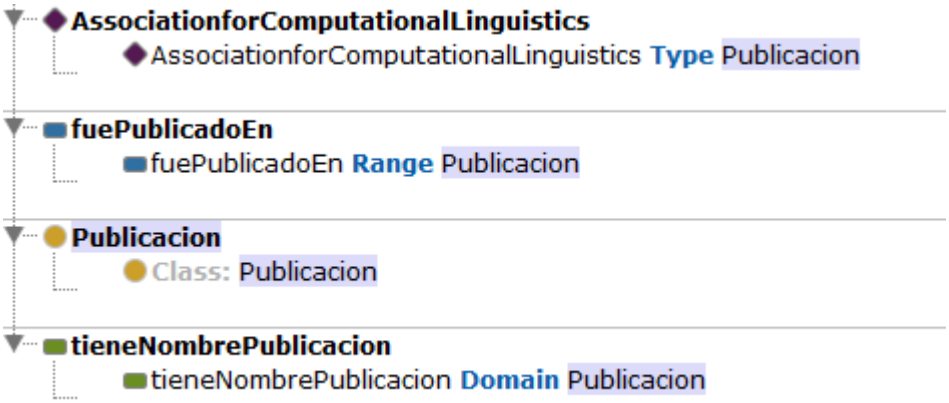


Figura 28. Propiedades de la clase Publicación

7. Resultados

Se realizó la búsqueda de 5 autores, las cuales se eligieron 2 publicaciones por cada uno para la carga de la ontología, la cual nos brindó un catálogo sólido de Instituciones, Artículos, Autores y Publicaciones, las cuales ya sólo se concentra en un solo ente principal, ya que si se introduce otro autor que trabajó en la mis institución o en la misma revista, etc. No se crea duplicidad pero sí se crean las relaciones ontológicas para poder hacer una extracción más a fondo de la información.
Como se muestra en la siguiente tabla, podemos que se pudo obtener un 85% de la información que se pudo sacar del recorrido de la búsqueda, el 15% restante se debe a que algunos autores no llenan la información completa y esto la hace incompleta.
Con esto podemos considerar que la aplicación realizó una búsqueda con éxito para poder poblar las ontologías necesarias.

8. Conclusiones

Este proyecto utiliza un sistema de búsqueda en la web se Google Scholar mediante etiquetas para sólo obtener la información que se necesita, por ello nos ahora mucho el procesamiento de la información y obtener información puntual.
Las reglas de relaciones ontológicas creadas nos ayudan a tener un banco de información más precisa, con información confiable y con los temas de gran interés, con la evolución de esta ontología nos permite obtener nuevas abstracciones de información, como obtener rápidamente relaciones de proyectos por temas o por autores, que instituciones produce más investigación sobre cierto tema, cuales son las orientaciones de publicaciones de las revistas por los temas de investigación de los autores que trabajan con ellos.
Con todo lo anterior podemos observar que el crecimiento de esta ontología con todos los datos que encuentra la aplicación nos mostrará un gran banco de información, así como nueva abstracción de información.
Este proyecto tiene un gran futuro como buscador y extractor de información que se puede llegar a incluir nuevos motores de búsquedas para poder tener una visión más amplia de la información, así como también poder tener datos más puntuales sobre lo que estamos buscando.

9. Referencias Bibliográficas:

[1] C. M. Pilapanta Herrera, Poblados Automático de Ontologías, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2015.

[2] I. A. Rosas Torres, Sistema Web para la identificación automática de aspectos académicos y de experiencia profesional en expedientes curriculares, Universidad Autónoma Metropolitana Azcapotzalco, México, 2015.

[3] F.A Gudiño Pérez, Representación semántica y extracción de información sobre publicaciones en expedientes curriculares, Universidad Autónoma Metropolitana Azcapotzalco, México, 2014.

[4] A. García García, Datos abiertos enlazados Linked Open Data (LOD) en Documentación Científica, https://riunet.upv.es/handle/10251/18272, 2013.

[5] J. A. Pastor Sanchez, Ontologías de control de autoridades en el ámbito de los datos abiertos enlazados, http://cat.inist.fr/?aModele=afficheN&cpsidt=28010672, 2013.

10. Anexos

Anexo A1. Obtener información de autores relacionados a la búsqueda.

```
import java.io.IOException;
import java.util.ArrayList;
import objetos.Autor;
import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.slf4j.LoggerFactory;
public class BuscarAutor {
	static final org.slf4j.Logger logger = LoggerFactory.getLogger(BuscarAutor.class);
	//Liga que se concatena con el nombre del autor a buscar
		private static final String GCR_SEARCH_THSIS_URL =
"https://scholar.google.com/citations?view_op=search_authors&mauthors=";
		//Etiqueta del bloque de los autores encontrados
		private static final String THSIS_ELEMENT_CLASS = "gsc_1usr_text";
		//Etiquetas con los valores que se desea obtener
	private static final String THSIS_Nombre = "gsc_1usr_name";
	private static final String THSIS_Titulo = "gsc_1usr_aff";
	private static final String THSIS_Correo = "gsc_1usr_eml";
		private static final String THSIS_Temas = "gsc_co_int";
	private static final String THSIS_Url = "href";
		private BuscarAutor() {
		}

		public static Document connect(String url){
			logger.info("info");
			Connection conn = Jsoup.connect(url);
			Document document = null;
			conn.header("Referer", "http://scholar.google.com/");
			conn.userAgent("Mozilla/17.0 (compatible; MSIE 6.0; Windows NT 5.0)");
			conn.timeout(3*1000);
			conn.method(Connection.Method.POST);
			try{
				Connection.Response response = conn.execute();
				int statusCode = response.statusCode();
				if(statusCode == 200) {
					document = conn.get();
					logger.info("info");
				}
				else {
				   System.out.println("received error code : " + statusCode);
				}
			}catch(IOException ioe){
			}
			return document;
```

```
	}
	public static ArrayList<Autor> BuscarAutores(String aut) throws IOException {
		ArrayList<Autor> autores = new ArrayList<>();
		String encodedThsisTitle = aut.trim().replace(" ", "+");
		String url = GCR_SEARCH_THSIS_URL + encodedThsisTitle;
		Document htmldoc = connect(url);
		if(htmldoc == null){
			return null;
		}
		Elements thsisElements = htmldoc
				.getElementsByClass(THSIS_ELEMENT_CLASS);

	 Elements thsisElements2 = htmldoc
				.getElementsByClass(THSIS_Nombre);
	 int cont = 0;
		for (Element e : thsisElements) {
			Autor t = new Autor();
	      t.setTitulo(getTitulo(e));
	      t.setTemas(getTemas(e));
	      t.setNombre(getNombre(e));
	      t.setCorreo(getCorreo(e));
	      t.setUrl(thsisElements2.get(cont).childNodes().get(0).attributes().get("href"));
			autores.add(t);
	      cont++;
		}
		return autores;
	}
	private static String getTitulo(Element e) {
		return e.getElementsByClass(THSIS_Titulo).text();
	}

    private static String getTemas(Element e) {
		return e.getElementsByClass(THSIS_Temas).text();
	}

	private static String getCorreo(Element e) {
		return e.getElementsByClass(THSIS_Correo).text();
	}
	private static String getNombre(Element e) {
		return e.getElementsByClass(THSIS_Nombre).text();
	}
}
```

Anexo A2. Clase “Autor”

```
package objetos;

public class Autor {
```

```
    private String nombre;
    private String titulo;
    private String correo;
    private String temas;
    private String url;

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }

    public String getTemas() {
        return temas;
    }

    public void setTemas(String temas) {
        this.temas = temas;
    }

}
```

Anexo A3. Obtener información publicaciones, artículos del autor seleccionado.

```
import java.io.IOException;
import java.util.ArrayList;
import objetos.PublicacionAutor;
import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.slf4j.LoggerFactory;
public class DescripcionAutor {

   static final org.slf4j.Logger logger = LoggerFactory.getLogger(DescripcionAutor.class);
      //Liga que se concatena con el nombre del autor a buscar su informaciόm complementaria
          private static final String GCR_SEARCH_THSIS_URL = "https://scholar.google.com";
          //Lista de articulos encontrados
          private static final String THSIS_ELEMENT_CLASS = "gsc_a_tr";
          //Etiquetas con los valores que se desea obtener
      private static final String THSIS_Titulo = "gsc_a_at";
      private static final String THSIS_Colaboradores = "gs_gray";
          private DescripcionAutor() {
          }
          public static Document connect(String url){
                    logger.info("info");
                    Connection conn = Jsoup.connect(url);
                    Document document = null;
                    conn.header("Referer", "http://scholar.google.com/");
                    conn.userAgent("Mozilla/17.0 (compatible; MSIE 6.0; Windows NT 5.0)");
                    conn.timeout(3*1000);
                    conn.method(Connection.Method.POST);
                    try{
                              Connection.Response response = conn.execute();
                              int statusCode = response.statusCode();
                              if(statusCode == 200) {
                                        document = conn.get();
                                        logger.info("info");
                              }
                              else {
                                 System.out.println("received error code : " + statusCode);
                              }
                    }catch(IOException ioe){
                    }

                    return document;
          }
          public static ArrayList<PublicacionAutor> BuscarPublicacionAutor(String aut) throws
IOException {
                    ArrayList<PublicacionAutor> publicacionAutor = new ArrayList<>();
```

```
                String url = GCR_SEARCH_THSIS_URL + aut;
                Document htmldoc = connect(url);
                if(htmldoc == null){
                        return null;
                }
                Elements thsisElements = htmldoc
                                .getElementsByClass(THSIS_ELEMENT_CLASS);

        Elements thsisElements2 = htmldoc
                                .getElementsByClass(THSIS_Titulo);
        int cont = 0;
                for (Element e : thsisElements) {
                        PublicacionAutor t = new PublicacionAutor();
            t.setTitulo(getTitulo(e));
            t.setColaboradores(getColaboradores(e));
            t.setUrl(thsisElements2.get(cont).parent().childNode(0).attributes().get("href"));
                        publicacionAutor.add(t);
            cont++;
                }
                return publicacionAutor;
        }
        private static String getTitulo(Element e) {
                return e.getElementsByClass(THSIS_Titulo).text();
        }

    private static String getColaboradores(Element e) {
                return e.getElementsByClass(THSIS_Colaboradores).text();
        }
}
```

Anexo A4. Clase “PublicacionAutor”

```
package objetos;

public class PublicacionAutor {
   private String titulo;
   private String colaboradores;
   private String contacto;
   private String url;

   public String getUrl() {
      return url;
   }

   public void setUrl(String url) {
      this.url = url;
   }

   public String getTitulo() {
```

```
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getColaboradores() {
        return colaboradores;
    }

    public void setColaboradores(String colaboradores) {
        this.colaboradores = colaboradores;
    }

    public String getContacto() {
        return contacto;
    }

    public void setContactor(String contactor) {
        this.contacto = contactor;
    }
}
```

Anexo A5. Obtener datos del artículo seleccionado

```
import static controlador.DescripcionAutor.logger;
import java.io.IOException;
import objetos.ArticuloO;
import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.select.Elements;

public class DescripcionArticulo {

    private static final String GCR_SEARCH_THSIS_URL = "https://scholar.google.com";

        private static final String THSIS_ELEMENT_CLASS = "gsc_value";
    private static final String THSIS_Contactor = "gsc_field";
        private DescripcionArticulo() {
        }

        public static Document connect(String url){
                logger.info("info");
                Connection conn = Jsoup.connect(url);
                Document document = null;
                conn.header("Referer", "http://scholar.google.com/");
                conn.userAgent("Mozilla/17.0 (compatible; MSIE 6.0; Windows NT 5.0)");
```

```
            conn.timeout(3*1000);
            conn.method(Connection.Method.POST);
            try{
                    Connection.Response response = conn.execute();
                    int statusCode = response.statusCode();
                    if(statusCode == 200) {
                            document = conn.get();
                            logger.info("info");
                    }
                    else {
                       System.out.println("received error code : " + statusCode);
                    }
            }catch(IOException ioe){
            }

            return document;
    }

    public static ArticuloO BuscarArticulo(String aut) throws IOException {
            String url = GCR_SEARCH_THSIS_URL + aut;
            Document htmldoc = connect(url);
            if(htmldoc == null){
                    return null;
            }
            Elements thsisElements = htmldoc
                            .getElementsByClass(THSIS_ELEMENT_CLASS);

     Elements thsisElements2 = htmldoc
                            .getElementsByClass(THSIS_Contactor);
                    ArticuloO t = new ArticuloO();
          int cont = 0;
          String descripcion = "";
             for (int i = 0; i < thsisElements2.size(); i++){
                String e=thsisElements2.get(i).childNodes().get(0).toString();
             if(e.equalsIgnoreCase("\nAuthors")){
                t.setColaboradores(thsisElements.get(i).childNode(0).toString());
             }else if(e.equalsIgnoreCase("\nPublication date")){
                t.setFecha(thsisElements.get(i).childNode(0).toString());
             }else if(e.equalsIgnoreCase("\nJournal")){
                t.setJournal(thsisElements.get(i).childNode(0).toString());
             }else if(e.equalsIgnoreCase("\nVolume")){
                t.setVolumen(thsisElements.get(i).childNode(0).toString());
             }else if(e.equalsIgnoreCase("\nIssue")){
                t.setIssue(thsisElements.get(i).childNode(0).toString());
             }else if(e.equalsIgnoreCase("\nPages")){
                t.setPages(thsisElements.get(i).childNode(0).toString());
             }else if(e.equalsIgnoreCase("\nPublisher")){
                t.setPublisher(thsisElements.get(i).childNode(0).toString());
```

```
                    }else if(e.equalsIgnoreCase("\nDescription")){
                       int r = thsisElements.get(i).childNodeSize()-1;
                       for (int j = 0; j < r; j=j+2){
                          descripcion += " "+thsisElements.get(i).childNode(j).toString();
                       }
                       t.setDescription(descripcion);
                    }

                 }
                    return t;
           }
}
```

Anexo A6. Búsqueda por nombre de publicación.

```
import java.io.IOException;
import java.util.ArrayList;
import objetos.Publicacion;
import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.slf4j.LoggerFactory;

public class BuscarArticulo {
   static final org.slf4j.Logger logger = LoggerFactory.getLogger(BuscarArticulo.class);

           private static final String GCR_SEARCH_THSIS_URL =
"https://scholar.google.com.mx/scholar?hl=es&q=";

           private static final String THSIS_ELEMENT_CLASS = "gs_ri";
       private static final String THSIS_Titulo = "gs_rt";
       private static final String THSIS_Colaboradores = "gs_a";
       private static final String THSIS_Resumen = "gs_rs";
           private static final String THSIS_Url = "gs_ctc";


           private BuscarArticulo() {
           }

           public static Document connect(String url){
                    logger.info("info");
                    Connection conn = Jsoup.connect(url);
                    Document document = null;
                    conn.header("Referer", "http://scholar.google.com/");
                    conn.userAgent("Mozilla/17.0 (compatible; MSIE 6.0; Windows NT 5.0)");
                    conn.timeout(3*1000);
                    conn.method(Connection.Method.POST);
```

```
		try{
			Connection.Response response = conn.execute();
			int statusCode = response.statusCode();
			if(statusCode == 200) {
				document = conn.get();
				logger.info("info");
			}
			else {
			   System.out.println("received error code : " + statusCode);
			}
		}catch(IOException ioe){
		}

		return document;
	}

	public static ArrayList<Publicacion> BuscarPublicacion(String aut) throws IOException {
		ArrayList<Publicacion> publicacion = new ArrayList<Publicacion>();
	 String encodedThsisTitle = aut.trim().replace(" ", "+");
		String url = GCR_SEARCH_THSIS_URL + encodedThsisTitle;
		Document htmldoc = connect(url);
		if(htmldoc == null){
			return null;
		}
		Elements thsisElements = htmldoc
				.getElementsByClass(THSIS_ELEMENT_CLASS);

	 Elements thsisElements2 = htmldoc
				.getElementsByClass(THSIS_Titulo);
	 int cont=0;
		for (Element e : thsisElements) {
			Publicacion t = new Publicacion();
	    t.setTitulo(getTitulo(e));
	    t.setColaboradores(getColaboradores(e));
	    t.setResumen(getResumen(e));
	    t.setUrl(thsisElements2.get(cont).childNodes().get(0).attributes().get("href"));
			publicacion.add(t);
	    cont++;
		   }
		   return publicacion;
	}

	private static String getTitulo(Element e) {
		return e.getElementsByClass(THSIS_Titulo).text();
	}

private static String getColaboradores(Element e) {
		return e.getElementsByClass(THSIS_Colaboradores).text();
```

```
        }

        private static String getResumen(Element e) {
                return e.getElementsByClass(THSIS_Resumen).text();
        }
    private static String getUrl(Element e) {
                return e.getElementsByClass(THSIS_Url).text();
        }

}
```

Anexo A7. Clase “ArticuloO”

```
package objetos;

public class ArticuloO {

   private String colaboradores;
   private String fecha;
   private String journal;
   private String volumen;
   private String issue;
   private String pages;
   private String publisher;
   private String description;

   public String getColaboradores() {
      return colaboradores;
   }

   public void setColaboradores(String colaboradores) {
      this.colaboradores = colaboradores;
   }

   public String getFecha() {
      return fecha;
   }

   public void setFecha(String fecha) {
      this.fecha = fecha;
   }

   public String getJournal() {
      return journal;
   }

   public void setJournal(String journal) {
      this.journal = journal;
   }
```

```
    public String getVolumen() {
        return volumen;
    }

    public void setVolumen(String volumen) {
        this.volumen = volumen;
    }

    public String getIssue() {
        return issue;
    }

    public void setIssue(String issue) {
        this.issue = issue;
    }

    public String getPages() {
        return pages;
    }

    public void setPages(String pages) {
        this.pages = pages;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

}
```

Anexo B1. Creación y carga de datos a ontología.

```
package ontologia;

import java.io.File;
import objetos.ArticuloO;
```

```
import objetos.Autor;
import objetos.PublicacionAutor;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.AddAxiom;
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.OWLDataPropertyAssertionAxiom;
import org.semanticweb.owlapi.model.OWLIndividual;
import org.semanticweb.owlapi.model.OWLObjectProperty;
import org.semanticweb.owlapi.model.OWLObjectPropertyAssertionAxiom;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.model.OWLOntologyStorageException;



public class CargaOntologia {

   private static OWLOntologyManager manejador = null;
   private static OWLOntology ontologia;
   private static OWLDataFactory factory;
   private static OWLObjectProperty opEstaAdscritoA;
   private static OWLObjectProperty opFuePublicadoEn;
   private static OWLObjectProperty opFuePublicadoPor;
   private static OWLDataProperty dpTieneAnioDePublicacion;
   private static OWLDataProperty dpTieneAreaInvestigacion;
   private static OWLDataProperty dpTieneCorreo;
   private static OWLDataProperty dpTieneNombreInstitucion;
   private static OWLDataProperty dpTieneNombreInvestigador;
   private static OWLDataProperty dpTieneNombrePublicacion;
   private static OWLDataProperty dpTienePagInicial;
   private static OWLDataProperty dpTienePosicionAcademica;
   private static OWLDataProperty dpTieneTitulo;
   private OWLClass claseArticulo;
   private OWLClass claseInstitucion;
   private OWLClass claseInvestigador;
   private OWLClass clasePublicacion;

   private static final File direccion = new File("D:\\ProyrectoTerm\\OntologiaAcademica.owl");
   private static final String url
="http://www.semanticweb.org/alumno/ontologies/2016/0/OntologiaAcademica";

public void leerOntologia() throws OWLOntologyStorageException,
OWLOntologyCreationException {
```

```
        manejador = OWLManager.createOWLOntologyManager();
        ontologia = manejador.loadOntologyFromOntologyDocument(direccion);

        factory = manejador.getOWLDataFactory();

        //Obtenemos las clases de la ontolog﷿a
        claseArticulo = factory.getOWLClass(IRI.create(url + "#Articulo"));
        claseInstitucion = factory.getOWLClass(IRI.create(url + "#Institucion"));
        claseInvestigador = factory.getOWLClass(IRI.create(url + "#Investigador"));
        clasePublicacion = factory.getOWLClass(IRI.create(url + "#Publicacion"));


        //Obtenemos los ObjectProperty
        opEstaAdscritoA = factory.getOWLObjectProperty(IRI.create(url + "#estaAdscritoA"));
        opFuePublicadoEn = factory.getOWLObjectProperty(IRI.create(url + "#fuePublicadoEn"));
        opFuePublicadoPor = factory.getOWLObjectProperty(IRI.create(url + "#fuePublicadoPor"));


        //Obtenemos los DataProperty
        dpTieneAnioDePublicacion = factory.getOWLDataProperty(IRI.create(url +
"#tieneAnioDePublicacion"));
        dpTieneAreaInvestigacion= factory.getOWLDataProperty(IRI.create(url +
"#tieneAreaInvestigacion"));
        dpTieneCorreo = factory.getOWLDataProperty(IRI.create(url + "#tieneCorreo"));
        dpTieneNombreInstitucion = factory.getOWLDataProperty(IRI.create(url +
"#tieneNombreInstitucion"));
        dpTieneNombreInvestigador = factory.getOWLDataProperty(IRI.create(url +
"#tieneNombreInvestigador"));
        dpTieneNombrePublicacion = factory.getOWLDataProperty(IRI.create(url +
"#tieneNombrePublicacion"));
        dpTienePagInicial = factory.getOWLDataProperty(IRI.create(url + "#tienePagInicial"));
        dpTienePosicionAcademica = factory.getOWLDataProperty(IRI.create(url +
"#tienePosicionAcademica"));
        dpTieneTitulo = factory.getOWLDataProperty(IRI.create(url + "#tieneTitulo"));

    }
    public void cargarAutor(Autor autor,PublicacionAutor publicacion,ArticuloO arti) throws
OWLOntologyStorageException{

        OWLIndividual owlAutor = factory.getOWLNamedIndividual(IRI.create(url + "#" +
autor.getNombre().replaceAll(" ", "")));
        OWLIndividual owlInstitucion = factory.getOWLNamedIndividual(IRI.create(url + "#" +
autor.getTitulo().replaceAll(" ", "")));
        OWLIndividual owlArticulo = factory.getOWLNamedIndividual(IRI.create(url + "#" +
publicacion.getTitulo().replaceAll(" ", "")));
        OWLIndividual owlPublicacion = factory.getOWLNamedIndividual(IRI.create(url + "#" +
arti.getPublisher().replaceAll("\n", "").replaceAll(" ", "")));
```

```
    OWLDataPropertyAssertionAxiom correo =
factory.getOWLDataPropertyAssertionAxiom(dpTieneCorreo, owlAutor,
autor.getCorreo().replaceAll("Verified email at ",""));
    OWLDataPropertyAssertionAxiom temas =
factory.getOWLDataPropertyAssertionAxiom(dpTieneAreaInvestigacion, owlAutor,
autor.getTemas());
    OWLDataPropertyAssertionAxiom nombre =
factory.getOWLDataPropertyAssertionAxiom(dpTieneNombreInvestigador, owlAutor,
autor.getNombre());
    OWLDataPropertyAssertionAxiom titulo =
factory.getOWLDataPropertyAssertionAxiom(dpTienePosicionAcademica, owlAutor,
autor.getTitulo());
    OWLDataPropertyAssertionAxiom nomIStitu =
factory.getOWLDataPropertyAssertionAxiom(dpTieneNombreInstitucion, owlInstitucion,
autor.getTitulo());
    OWLDataPropertyAssertionAxiom anio =
factory.getOWLDataPropertyAssertionAxiom(dpTieneAnioDePublicacion, owlArticulo,
arti.getFecha());
    OWLDataPropertyAssertionAxiom pag =
factory.getOWLDataPropertyAssertionAxiom(dpTienePagInicial, owlArticulo, arti.getPages());
    OWLDataPropertyAssertionAxiom art =
factory.getOWLDataPropertyAssertionAxiom(dpTieneTitulo, owlArticulo,
publicacion.getTitulo());
    OWLDataPropertyAssertionAxiom publi =
factory.getOWLDataPropertyAssertionAxiom(dpTieneNombrePublicacion, owlPublicacion,
arti.getPublisher().replaceAll("\n", ""));


    OWLObjectPropertyAssertionAxiom AutorIntsti =
factory.getOWLObjectPropertyAssertionAxiom(opEstaAdscritoA, owlAutor, owlInstitucion);
    OWLObjectPropertyAssertionAxiom ArtiInves =
factory.getOWLObjectPropertyAssertionAxiom(opFuePublicadoPor, owlArticulo, owlAutor);
    OWLObjectPropertyAssertionAxiom Publica =
factory.getOWLObjectPropertyAssertionAxiom(opFuePublicadoEn, owlArticulo,
owlPublicacion);


    AddAxiom addAxiomCorreo = new AddAxiom(ontologia, correo);
    AddAxiom addAxiomTemas = new AddAxiom(ontologia, temas);
    AddAxiom addAxiomNombre = new AddAxiom(ontologia, nombre);
    AddAxiom addAxiomTitulo = new AddAxiom(ontologia, titulo);
    AddAxiom addAxiomNomInsti = new AddAxiom(ontologia, nomIStitu);
    AddAxiom addAxiomAutorInstitu = new AddAxiom(ontologia, AutorIntsti);
    AddAxiom addAxiomAnio = new AddAxiom(ontologia, anio);
    AddAxiom addAxiomPag = new AddAxiom(ontologia, pag);
    AddAxiom addAxiomArt = new AddAxiom(ontologia, art);
    AddAxiom addAxiomArtiInves = new AddAxiom(ontologia, ArtiInves);
    AddAxiom addAxiomPubl = new AddAxiom(ontologia, publi);
```

```
        AddAxiom addAxiomArtPubl = new AddAxiom(ontologia, Publica);
        manejador.applyChange(new AddAxiom(ontologia,
factory.getOWLClassAssertionAxiom(claseInvestigador, owlAutor)));
        manejador.applyChange(new AddAxiom(ontologia,
factory.getOWLClassAssertionAxiom(claseInstitucion, owlInstitucion)));
        manejador.applyChange(new AddAxiom(ontologia,
factory.getOWLClassAssertionAxiom(claseArticulo, owlArticulo)));
        manejador.applyChange(new AddAxiom(ontologia,
factory.getOWLClassAssertionAxiom(clasePublicacion, owlPublicacion)));
        manejador.applyChange(addAxiomCorreo);
        manejador.applyChange(addAxiomTemas);
        manejador.applyChange(addAxiomNombre);
        manejador.applyChange(addAxiomTitulo);
        manejador.applyChange(addAxiomNomInsti);
        manejador.applyChange(addAxiomAutorInstitu);
        manejador.applyChange(addAxiomAnio);
        manejador.applyChange(addAxiomPag);
        manejador.applyChange(addAxiomArt);
        manejador.applyChange(addAxiomArtiInves);
        manejador.applyChange(addAxiomPubl);
        manejador.applyChange(addAxiomArtPubl);
        manejador.saveOntology(ontologia,manejador.getOntologyDocumentIRI(ontologia));

    }
}
```

11. Manual de Instalación

Índice General

Índice de Figuras

1. Descripción de archivos
    a. La carpeta PublicacionesOntologías, contiene el código fuente de la aplicación
    b. Carpeta Ontología, contiene la OWL OntologiaAcademica para cargar las ontologías obtenidas por la aplicación.
    c. Carpeta Librerías, contiene los jar necesarios para poder compilar el proyecto.
    d. Carpeta Ejecutable, contiene el jar del proyecto.
2. Requisitos del sistema: para la ejecución del sistema se deben contar con los siguientes componentes:
    a. Un sistema operativo recomendado Windows 7 (o posterior) o Linux (Distribuciones como Ubuntu o Debian).
    b. Un entorno de desarrollo IDE compatible con lenguaje Java, se recomienda: NetBeans: es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.
    c. Protégé: es un editor de ontologías de código libre que sirve para modelar y cargar datos a una ontología.
3. Preparando el sistema
    a. Creando carpta para la ontología: para la lectura y escritura de la ontología es necesario crear un carpeta en la siguiente ruta: “D:\ProyectoTermnial”.
    b. Copiar los archivos que se encuentran en la ruta: “D:\ProyrectoTerm\ArticuloOntologia\dist” en el lugar que prefiera.
    c. Descargar Protégé de la siguiente liga: “http://protege.stanford.edu/products.php#desktop-protege”
4. Ejecutar la aplicación: ArticuloOntologia que se encuentra en la ruta mencionada en el punto anterior.
    a. Intalar Protégé.
    b. Llenar datos del autor a buscar y hacer click en autor como se muestra en la Figura 1.

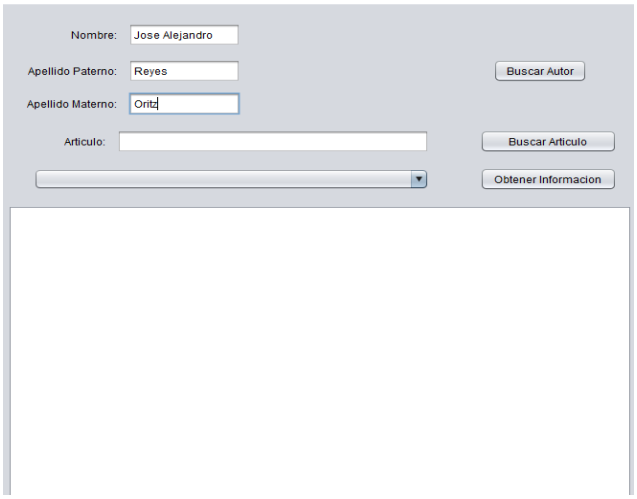


Figura 1. Imagen principal de la aplicación

c. Seleccionar el autor que se desea obtener su información y hacer click en ObtenerInformacion como se muestra en la Figura 2.

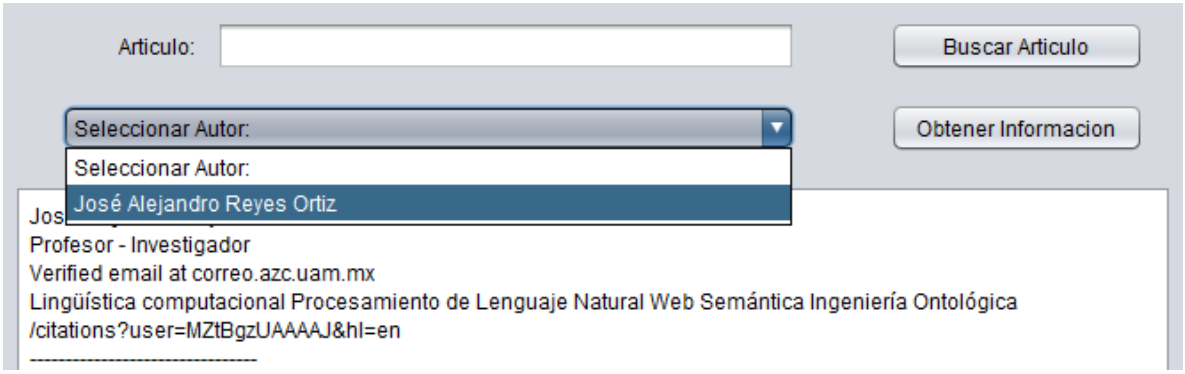


Figura 2. Lista de autores encontrados

d. Mostrará una nueva ventana con los datos del autor elegido y se procederá a elegir el artículo para luego hacer click en el botón Consultar Publicación como se muestra en la Figura 3.

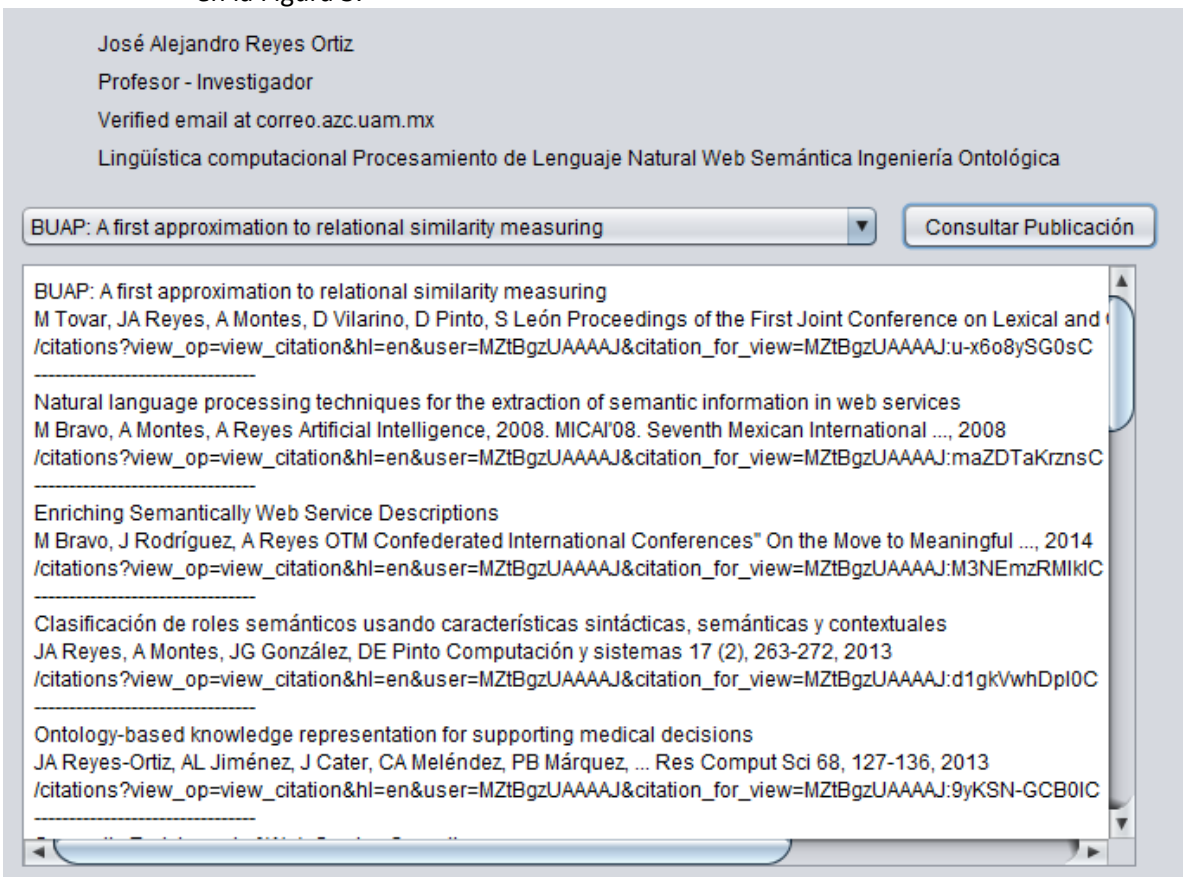


Figura 3. Obtención de información del autor

e. Luego se abrirá otra ventana con los datos del artículo seleccionado para luego hacer click en el botón Cargar Ontología para introducir los valores a la ontología como se muestra en la Figura 4.

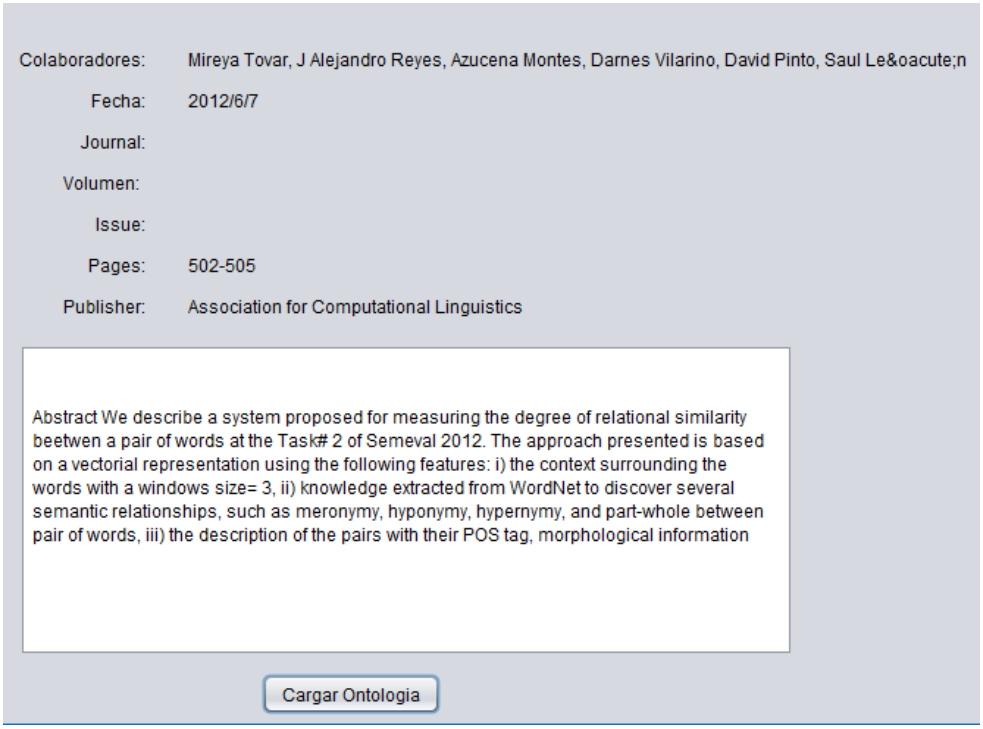


Figura 4. Información del artículo seleccionado

f. Para verificar que la carga de la ontología e realizó con éxito, pasamos a abrir el protegé y cargamos el archivo OntologiaAcademica.owl y debe aparecer la información como las figuras siguientes:

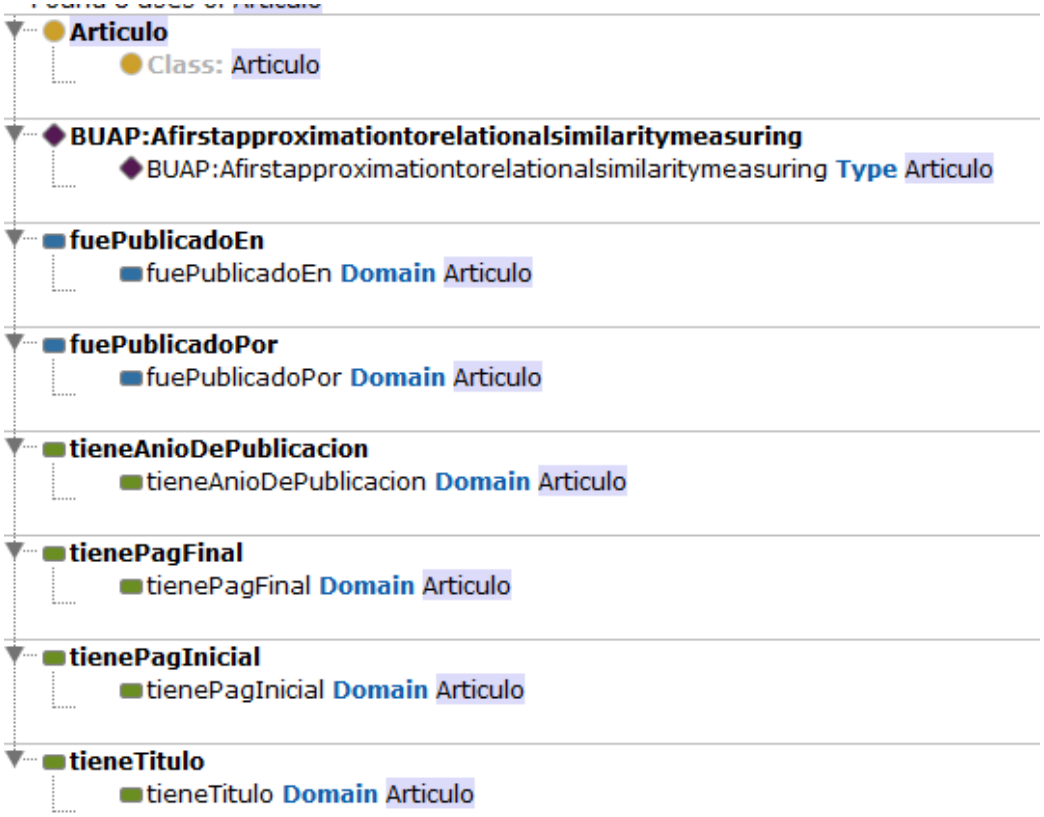


Figura 5. Propiedades de la clase Artículo

Figura 6. Propiedades de la clase Institucion

Figura 7. Propiedades de la clase Investigador

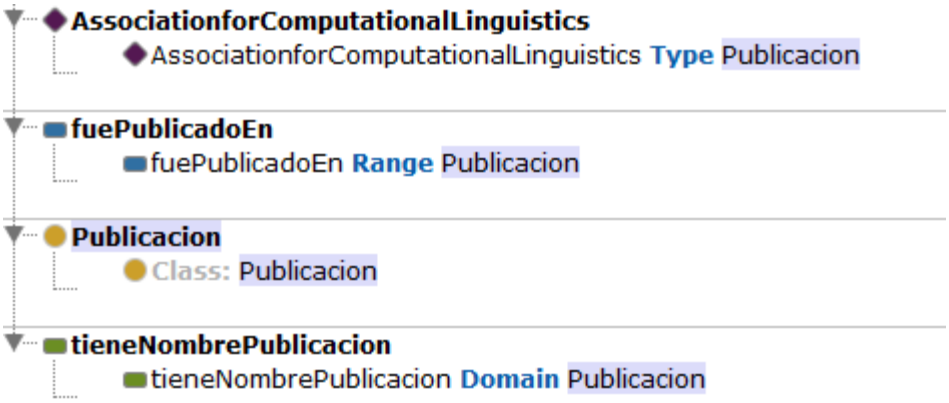


Figura 8. Propiedades de la clase Publicación