Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Sistema web para el poblado automático de ontologías a partir de textos

“Proyecto Tecnológico”

Carlos Mauricio Pilapanta Herrera
Matricula: 2113032396
al2113032396@alumnos.azc.uam.mx

Asesor:
Maricela Claudia Bravo Contreras
Profesor Asociado
Departamento de Sistemas
mcbc@correo.azc.uam.mx

Co asesor:
José Alejandro Reyes Ortiz
Profesor Titular
Departamento de sistemas
jaro@correo.azc.uam.mx

Trimestre 15-P
Fecha de entrega:
27 de julio de 2015

Yo, Maricela Claudia Bravo Contreras, 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, 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, Carlos Mauricio Pilapanta Herrera, 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

Uno de los problemas a los que enfrentan los analistas de noticias es tener que extraer la información de textos de manera manual. La información extraída es de gran utilidad para la toma de decisiones en materia de política, sociedad, seguridad, entre otros. Es por esta razón, que existe la necesidad de contar con herramientas informáticas de análisis de la información textual que disminuya en tiempo y costo invertido en esta tarea por parte de los humanos. Para ello, se realiza este proyecto con la finalidad de realizar la extracción de información de textos en PDF o textos (txt), tarea que no es trivial.

Este proyecto se apoya en librerías que permiten analizar, manipular y etiquetar el texto. Esto se lo realiza para poder tener la información ordenada, identificada y procesable por computadora permitiendo que usuarios o agentes realicen sus tareas de manera fácil y rápida.

Para poder solucionar el problema en este proyecto se plantea el uso de ontologías, ya que éstas permiten tener la información ordenada y con cierto significado (contenido semántico), la cual se puede consultar en cualquier momento. Para la creación de la ontología se utiliza una API de desarrollo que permite, procesar el texto etiquetado, diseñar la ontología y poblar la ontología para que pueda ser usada por investigadores o personas que necesiten información extraída de los textos de noticias.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Anexos

1. INTRODUCCIÓN

Al paso de los años el crecimiento de la información ha sido muy acelerada, es por eso que con este proyecto planea desarrollar un sistema web que nos permita procesar la información y almacenarla en una base de datos de ontologías ya que todos estos datos no cuentan con una estructura con la cual se puedan ver los datos, los cuales nos serán relevantes para el desarrollo de este proyecto, todo esto se realiza para que esta información posteriormente se pueda manipular por computadora.

La transformación del texto no es trivial ya que involucra conocimiento específico que debe estar representado en reglas, además representar esta información requiere de un mecanismo que proporcione semántica a la información, aplicando y desarrollado algoritmos usados en un lenguaje de procesamiento de textos.

Por lo tanto para el desarrollo de este sistema web hara uso de tecnologías como COnET que permite establecer ciertas reglas de procesamiento de textos, además con esto se podrá ayudar en gran medida al problema de la falta de estructura de la información.

Para ello, se propone un sistema web que convierta la información no estructurada de documentos electrónicos en ontologías que nos ayudan a dotar de significado y estructura a la información. Se utilizarán reglas para la extracción de información, y como resultado se obtendrán ontologías que estarán implementadas en OWL[1].

2. ANTECEDENTES

Proyectos Terminales o de Integración.

Sistema de procesamiento de textos de investigación [1]

Con este proyecto es posible procesar artículos de investigación que están escritos en inglés y que además están estructurados de acuerdo al formato, establecido por la IEEE[1], este sistema plantea la extracción de cierta información que es de interés para el investigador, como respuesta a esta necesidad, hoy en día ha surgido de entre las ramas de la inteligencia artificial y la lingüística computacional, el procesamiento del lenguaje natural, que mediante la implementación de sus técnicas permite a las máquinas manejar lenguajes no formales.

Representación semántica de información espacial y temporal a partir de textos periodísticos mediante reglas lingüísticas. [2]

Con este proyecto es posible procesar información, con este sistema se plantea extracción de información disponible en textos periodísticos escritos en lenguaje español. Con la ayuda de técnicas del procesamiento del lenguaje natural se permitirá identificar información relevante como el lugar y la fecha en la que se llevaron a cabo los hechos.

[1] **OWL:** Por sus siglas en inglés de Ontology Web Language- Lenguaje Web Ontológico.

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

Con este proyecto existe la posibilidad de poder realizar la extracción de información de publicaciones sobre expedientes curriculares para su almacenamiento y recuperación de la misma mediante ontologías. Con el sistema es posible extraer artículos y publicaciones de los expedientes utilizando técnicas de procesamiento del lenguaje natural.

Tesis.

Creación Automática de Ontologías a partir de textos con un enfoque Lingüístico. [4]

En esta tesis se utilizaron herramientas para el procesamiento del lenguaje, muy similares a las que en esta propuesta se plantean usar con la particularidad que la aplicación resultante no será una aplicación más bien es un módulo que permitirá realizar ciertas operaciones que son necesarias para el procesamiento, también servirá para establecer parámetros de referencia que nos permitirán avanzar en el desarrollo del mismo.

3. JUSTIFICACIÓN

Con el alto crecimiento de la información y la gran cantidad de la misma, su manejo, distribución y recuperación se vuelve un reto para las computadoras. Además, no se cuenta con un repositorio estructurado que contenga esta información, es por eso que este proyecto es necesario, por lo tanto este sistema web que nos ayudará a procesar la información y transformarla en un esquema estructurado semánticamente.

Para un mejor uso de la información es necesario crear reglas, estas reglas nos van a permitir establecer una relación y unificarla para obtener nuevos puntos de vista, al diseñar estas reglas se nos permitirá establecer un lenguaje común, el cual se podrá usar en múltiples contextos, las personas que podrán beneficiarse de nuestro sistema web son aquellos usuarios que necesitan tener la información dotada de un significado para obtener aquella información que ellos necesitan de manera precisa.

Este sistema utilizará reglas implementadas mediante expresiones regulares para la extracción de información de instancias y sus relaciones a partir de textos. La información extraída será poblada, es decir, instanciada en ontologías, las cuales son mecanismos que provienen de una rama de la inteligencia artificial además éstas utilizan XML[2] como su lenguaje de representación. Las ontologías se utilizan en la computación, por ello, este proyecto de integración está enmarcado dentro de la carrera de Ingeniería en computación como un sistema web que se llevará en el Grupo de Investigación de

[2]**XML**: eXtensible Markup Languaje (Lenguaje de marcas extensible).

Sistemas de Información del Departamento de Sistemas en la Universidad Autónoma Metropolitana.

4. Objetivos

a. Objetivo general:

Diseñar e implementar un sistema web para extraer información necesaria que nos permitirán desempeñar el poblado[3] automático de ontologías a partir de textos, a través de reglas basadas en expresiones regulares.

b. Objetivos Específicos

- Diseñar e implementar un conjunto de reglas basadas en expresiones regulares que se utilizarán para el descubrimiento de instancias y sus relaciones en los textos.
- Implementar la interfaz de programación y aplicaciones de la API de desarrollo COnET[4] para la extracción de información a partir textos.
- Diseñar e implementar un sistema web para realizar el proceso completo de instanciación de ontologías a partir de textos, mostrando el resultado de manera textual.

[3] Poblado es el proceso de instanciación de elementos en la ontología.

[4] **COnET**: Creación de Ontologías de Eventos a partir de Textos en Español

5. Marco Teórico.

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

5.1 El Lenguaje de Ontologías Web.

El Lenguaje de Ontologías Web (OWL) está diseñado para ser usado en aplicaciones que necesitan procesar el contenido de la información en lugar de únicamente representar información para los humanos. OWL facilita un mejor mecanismo de interpretabilidad de contenido Web proporcionando vocabulario adicional junto con una semántica formal.

5.2 Lenguaje HTML.

Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, videos, entre otros. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación

5.3 Jape[5] – Gate[6]

Este sistema de información permite visualizar las relaciones ontológicas que existe en la información que se extrajo del texto. Este sistema se encuentra disponible en algunas versiones, también disponible para otros sistemas operativos (Windows, Linux, Mac), este sistema usa algoritmos que permiten realizar la extracción de la información y mostrarla aunque para mostrar nuevamente esa información se deberá analizar el texto.

5.4 NetBeans.

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.

5.5 Tomcat

Apache Tomcat funciona como un contenedor de servlets desarrollado en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de Java Server Pages (JSP) de Oracle Corporation (aunque creado por Sun Microsystems).

[5] **JAPE:** es un transductor de estados finitos que opera sobre anotaciones basadas en expresiones regulares.

[6] **GATE:** Arquitectura General de Ingeniería de texto, es una plataforma utilizada para el procesamiento de lenguaje natural.

6 Desarrollo del proyecto

Para la implementación del proyecto se utilizó el lenguaje de programación Java por su portabilidad y sencillez, NetBeans como ambiente de desarrollo (IDE), Apache Tomcat como contenedor web con soporte de servlets y JSPs, un framework llamado OWL API para crear, manejar y obtener la información necesaria de ontologías OWL y Gumby como framework para el desarrollo de la interfaz operable para el usuario.

Se utilizan Ontologías con información sobre relaciones entre individuos, los cuales tienen un formato digital OWL con distintos dominios.

El proyecto consta de cuatro módulos, los cuales se muestran en la figura 1 y se describen a continuación.

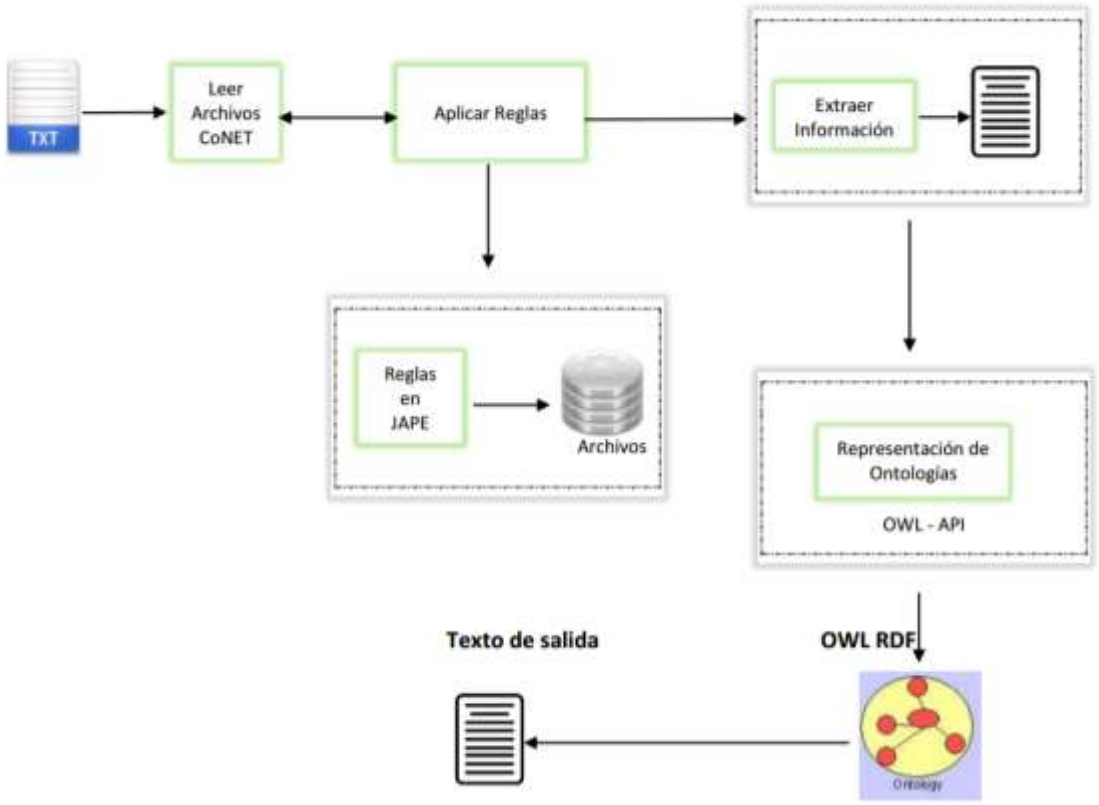

Figura 1. Arquitectura del sistema automático para la instanciación de ontologías.

6.1 Módulo Captura Petición del Usuario.

Este módulo permite al usuario subir los archivos de texto de tipo PDF y archivos de texto (.txt) al servidor para que pueda realizar la extracción de la información.

En la figura 2 se muestra la página principal de la aplicación, la cual es permite al usuario subir los archivos de texto. Además, este módulo también es el encargado de desplegar la página principal (index.jsp).

Fig. 2. Página principal de la aplicación

Para realizar la manipulación del archivo (PDF o txt), mediante código java a este archivo se lo convierte en una lista para que se pueda subir en partes, para lo cual se implementó un ciclo como se puede ver en la figura 3, el cual divide al archivo y lo vuelve a unir en la dirección final de servidor. Esta implementación se puede ver más a detalle en el anexo A1.

```
List<FileItem> partes = upload.parseRequest(request);
for (FileItem items : partes) {
    File file = new File(archivourl, items.getName());
    items.write(file);
    nombreDelTexto = items.getName();
}
```

Fig. 3. Algoritmo para procesamiento del archivo

Al dar clic en el botón se despliega una ventana en la que se puede elegir el tipo de archivo a procesar, el sistema tiene la capacidad de procesar archivos de texto en formato pdf y txt.

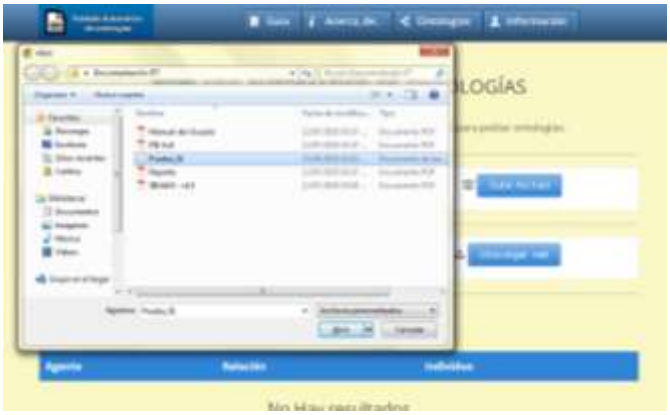

Fig. 4. Carga del archivo de texto a procesar

Una vez seleccionado el archivo, al presionar en el botón subir archivo, el modulo procesa el archivo, realiza una copia y la sube al servidor. Esto permite tener los recursos necesarios para que el sistema web pueda procesar la información y tenga un correcto desempeño. En la figura 5 se puede ver el archivo que se procesó.

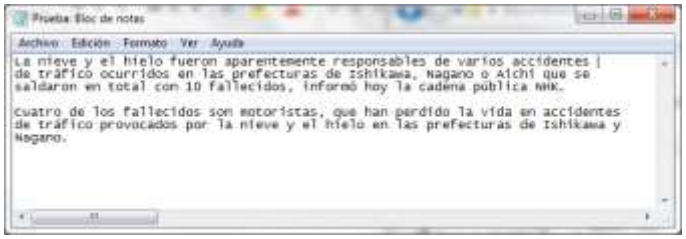

Fig. 5. Ejemplo de texto a procesar

Cuando el archivo se subió correctamente al servidor la página principal nos va a mostrar un mensaje de confirmación.

Fig. 6. Archivo subido correctamente

6.2 Módulo de Creación de reglas.

Este módulo permite implementar nuevas reglas que posteriormente se aplican en la etapa de extracción de la información, dichas reglas se almacenan en un archivo (.jape).

Las reglas codificadas con JAPE fueron creadas siguiendo un lenguaje de procesamiento verbal usado un software de procesamiento de texto llamado GATE en su septima versión. La tabla 1 muestra una regla de procesamiento verbal. El resto de las reglas se las puede ver más a detalle en el anexo B1.

Tabla 1. Sintagma Verbal

Reglas para la extracción de eventos sintagma verbal
SintagmaVerbal_1
Phase: PeriFrasis **Input: Token** **Options: control = brill** **Rule: SintagmaVerbal** **Priority: 20** **(** **(** **{Token.pos == VM} | {Token.pos == VS} | {Token.pos == VA}** **)** **(** **{Token.pos == AQ}** **)** **):pf -->** **:pf.SintagmaVerbal = {rule = "SintagmaVerbal",text =:pf@string}**

Las reglas implementadas se utilizan para la extracción de información que se presenta en la siguiente sección. En la figura 7 se muestra el código necesario para cargar de reglas, se usaron las siguientes instrucciones en código java. Las reglas son implementadas con la API GATE, la carga de reglas se puede observar a detalle en el anexo B2.

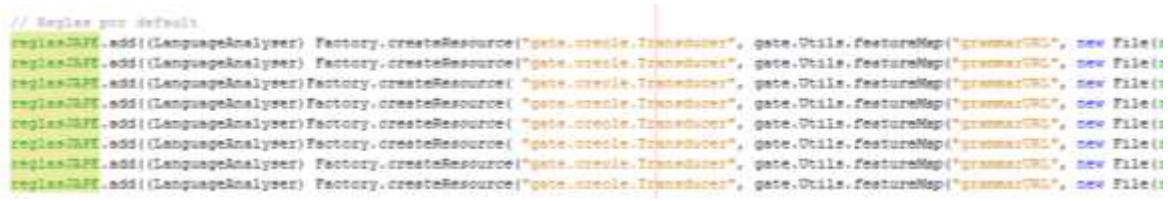

Fig. 7. Código para agregar las reglas y para aplicarlas

6.3 Módulo de Extracción Información.

En módulo permite aplicar las reglas que se crearon en el módulo anterior, que permiten extraer la información (frases verbales) que se requiere extraer de los textos (txt o pdf), esta información se la envía al siguiente modulo para su correspondiente representación. Esta información se puede observar de manera gráfica en la figura 8, cuyo texto se procesó en el software de procesamiento de textos GATE.

Fig. 8. Software Gate de procesamiento de texto

La extracción de información se implementó en java utilizando la API GATE, mediante la cual se procesa el texto que se cargó en el módulo de carga de archivos, el módulo del procesamiento divide el texto en tokens [7] en una primera pasada el texto, después se aplican las reglas que se crearon en módulo de creación de reglas. Para poder aplicar las reglas en lenguaje español se debe cargar plugins de procesamiento de lenguaje español, esta tarea se realiza con el siguiente código en lenguaje Java, Su implementación completa se la puede observar en el anexo C1.

```
// Registro del plugins
Gate.getCreoleRegister().registerDirectories(new File(rutaRelativa + "recursos/gate/plugins/SpanishPlugin").toURI().toURL());
Gate.getCreoleRegister().registerDirectories(new File(rutaRelativa + "recursos/gate/plugins/Tools").toURI().toURL());
Gate.getCreoleRegister().registerDirectories(new File(rutaRelativa + "recursos/gate/plugins/ANNIE").toURI().toURL());
```

Fig. 9. Código para agregar plugins.

La información extraída de los textos procesados al ser procesada se puede observar en la consola del IDE de programación netbeans tal como se muestra en la figura 10. EL texto etiquetado se almacena en dos listas que son regresadas al módulo de representación de resultados, para ver a detalle la implementación ver anexo C2.

```
log4j:WARN Please initialize the log4j system properly.
texto ok
{19=La_nieve_y_el_hielo, 96=varios_accidentes_de_tráfico_ocurridos, 115=las_prefecturas, 217=la_cadena_pública, 287=la_vida, 322=accidentes_de_tr
{168=saldaron, 194=informó, 228=Cuatro, 261=son_motoristas, 278=han_perdido}

Se creo la ontologia correctamente
```

Fig. 10. Texto procesado mostrado en la consola del IDE netbeans.

6.4 Módulo de Representación de Ontologías.

Este módulo recibe los parámetros generados por el módulo anterior, éstas salidas son los eventos verbales y los agentes, que pertenecen a la ontología, este es el modelo de representación de la información que es guardada en archivos con formatos rdf y owl,

[7] Token: Un token o también llamado componente léxico es una cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación.

dicha información es vista por el usuario en la página principal de nuestro sistema (index.jsp).

El sistema web “Poblado de Ontologías” procesa el texto y realiza el poblado de ontologías, este proceso se puede observar en la figura 11 además de la creación de la ontología con éxito que es usada en el siguiente módulo del sistema web. La implementación de la ontología se la puede ver más detalladamente en él anexo D1.

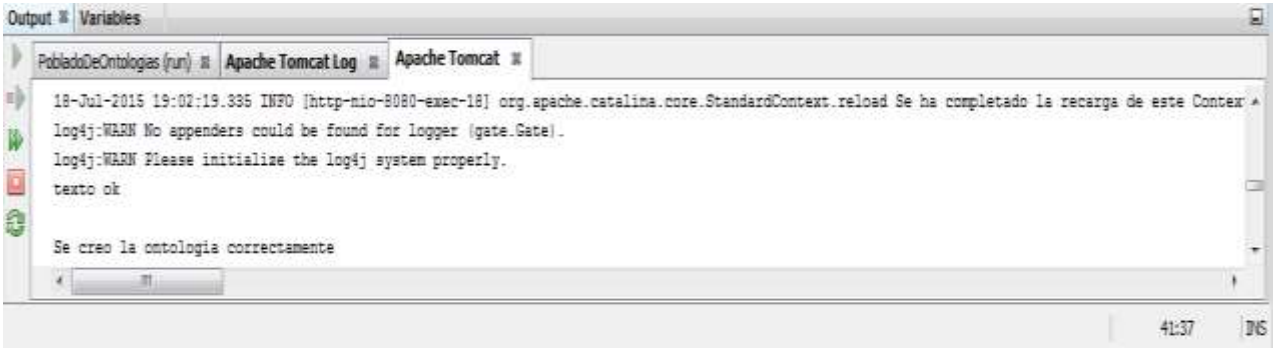

Fig 11. Inicio del procesamiento del texto mostrado en la consola del IDE netbeans.

Cuando se termina de procesar el texto, él sistema web, indica que el procesamiento termino con éxito, tal como se muestra en la siguiente figura 12.

Fig .12. Alerta de texto procesado con éxito.

Los resultados de la extracción de textos se pueden visualizar de manera textual en la página principal (index) como se puede observar en la figura 13. El código para la presentación de la ontología en él sistema web se puede ver a detalle en el anexo D2.

Fig 13. Resultados del procesamiento de texto.

La información extraída queda representada en una ontologías, la cual se puede ver de manera gráfica mediante el software llamado “Protege” que muestra la ontología con la estructura mostrada en la figura 14, y en la figura 15 se pueden ver los resultados de la población de ontologías en forma de árbol gracias al software antes mencionado.

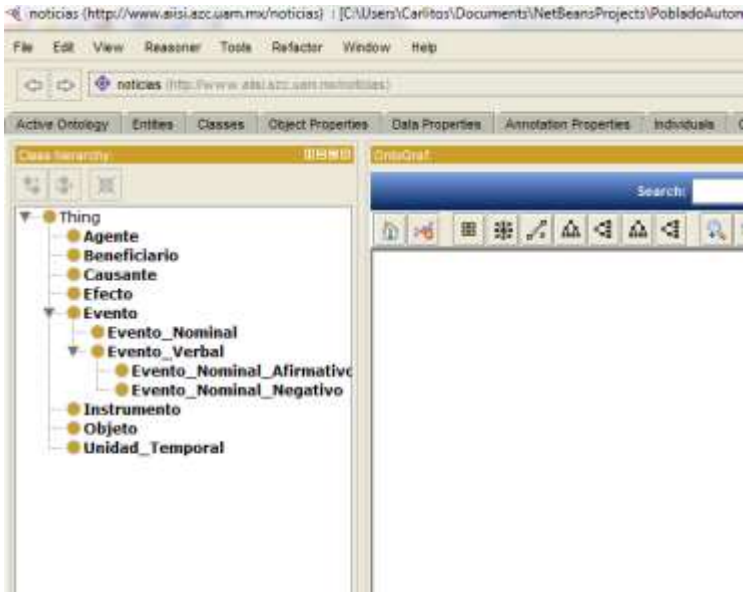

Fig. 14. Estructura de la ontología

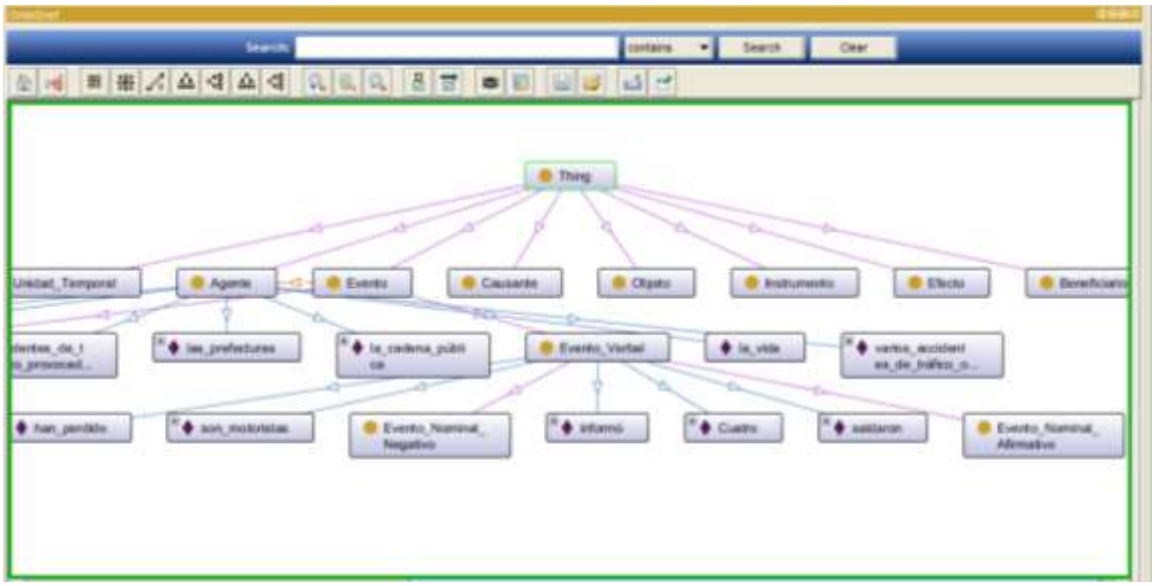

Fig. 15. Vista de la Ontología usando Protege.

7. Resultados

Para validar los resultados se hicieron pruebas en archivos de textos (.txt) con noticias. Que se muestran en la figura 16.

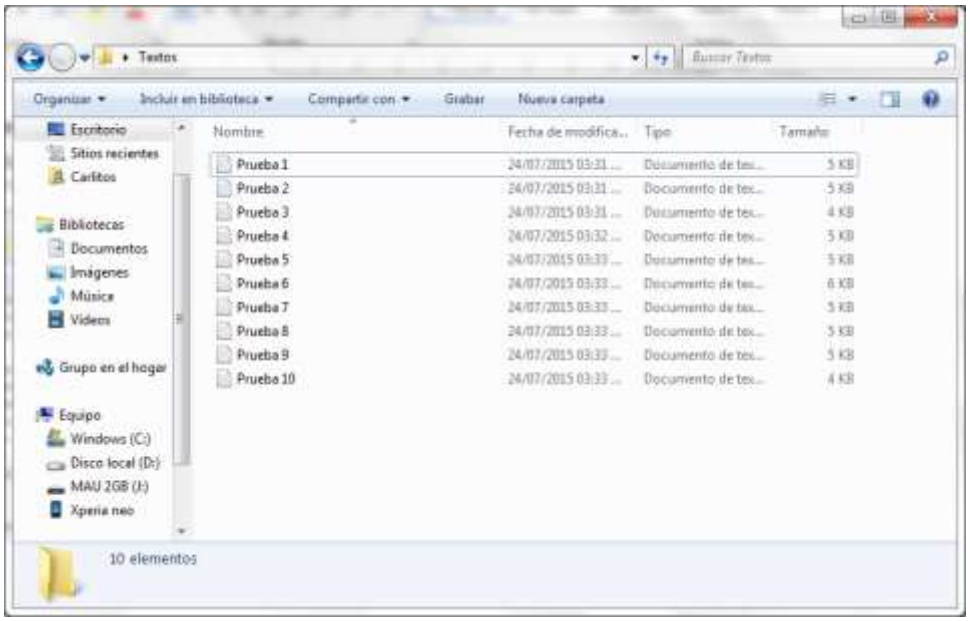

Fig. 16. Archivos para realizar pruebas.

Se realizaron un total de 10 pruebas, se obtuvieron los siguientes resultados mostrados en la tabla 2.

Tabla 2. Resultados de pruebas realizadas

Archivo	Sintagma Verbal	Frases Nominales	N° palabras en texto	% texto procesado
Prueba 1	70	119	753	25.10
Prueba 2	59	104	690	23.62
Prueba 3	62	103	674	24.48
Prueba 4	83	122	801	25.59
Prueba 5	62	124	810	22.96
Prueba 6	79	150	861	26.60
Prueba 7	69	109	658	27.05
Prueba 8	68	119	757	24.70
Prueba 9	1157	2773	17489	22.47
Prueba 10	1944	3215	20963	24.61

En base a los resultados obtenidos se puede decir que el procesamiento de texto tiene una recuperación del 25% aproximadamente, esta es información útil que sirve para poder poblar las ontologías.

8. Conclusiones

En este trabajo se presenta un sistema para el poblado automático de ontologías, usando API's de procesamiento de texto, utilizando textos de noticias y se crearon las ontologías, es decir, una base de datos que establecen distintos tipos de relaciones entre estos mismos que puede ser usado desde cualquier plataforma o sistema operativo siempre que tenga acceso al sistema web.

Con los resultados obtenidos, implementado este proyecto, se puede corroborar la funcionalidad de este sistema. Gracias a esto se puede poblar ontologías con textos de noticias con una efectividad aproximada del 25% de cualquier texto ingresado, esto es porque más del 75% del texto procesado no era necesario para los fines que se establecieron en u inicio, esto resulta un beneficio directo para los encargados de analizar la noticias ya que proporciona la información adecuada, además de beneficiar a los usuarios que desean localizar eventos o personas que realizaron cierta actividad.

9. Referencias bibliográficas

[1] Fernando Alejandro Acosta, “Sistemas de procesamiento de textos de investigación”, Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2013.

[2] Josué Padilla Cuevas, “Representación semántica de información espacial y temporal a partir de textos periodísticos mediante reglas lingüísticas”, Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2014.

[3]Francisco Alejandro Gudiño Pérez, “Representación semántica y extracción de información sobre publicaciones en expedientes curriculares”, Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2014.

[4] José Alejandro Reyes Ortiz, “Creación automática de ontologías a partir de textos con un enfoque lingüísticos”, Tesis de Doctorado en ciencias, Centro Nacional de Investigación y Desarrollo Tecnológico, México, 2013.

10. ANEXOS

Anexó A1. Carga del Archivo.

```
<%@page import="edu.conet.pln.principal.Principal"%>
<%@page import="org.apache.commons.fileupload.disk.DiskFileItemFactory"%>
<%@page import="org.apache.commons.fileupload.servlet.ServletFileUpload"%>
<%@page import="java.util.List"%>
<%@page import="org.apache.commons.fileupload.FileItem"%>
<%@page import="java.io.File"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%
    String archivourl = "C:/recursos/textos";
    String nombreDelTexto = " ";
    String resultado;
    DiskFileItemFactory factory = new DiskFileItemFactory();
    factory.setSizeThreshold(1024);
    factory.setRepository(new File(archivourl));
    ServletFileUpload upload = new ServletFileUpload(factory);
    try {
        List<FileItem> partes = upload.parseRequest(request);
        for (FileItem items : partes) {
            File file = new File(archivourl, items.getName());
            items.write(file);
            nombreDelTexto = items.getName();
        }
        resultado = "ok";
        request.getSession().setAttribute("nombreDelTexto", nombreDelTexto);
        request.getSession().setAttribute("resultado", resultado);
        response.sendRedirect("index.jsp");
    } catch (Exception e) {
        resultado = "fail";
        request.getSession().setAttribute("resultado", resultado);
        response.sendRedirect("index.jsp");
    }
%>
```

Anexo B1. Reglas para la extracción de textos.

Regla 1

```
Phase:  PeriFrasis
Input:  Token
Options: control = brill

Rule: SintagmaVerbal
Priority: 20
(
  (
    {Token.pos == VM} | {Token.pos == VS} | {Token.pos == VA}
  )
  (
    {Token.pos == AQ}
  )
):pf -->
:pf.SintagmaVerbal = {rule = "SintagmaVerbal",text =:pf@string}
```

Regla 2

```
Phase:  PeriFrasis
Input:  Token
Options: control = brill

Rule: SintagmaVerbal
Priority: 20
(
  (
    {Token.pos == VA} | {Token.pos == VS} | {Token.pos == VM}
  )
  (
    {Token.pos == SP} | {Token.pos == CC}
  )
  (
    {Token.pos == VM}
  )
):pf -->
:pf.SintagmaVerbal = {rule = "SintagmaVerbal",text =:pf@string}
```

Regla 3

```
Phase:  PeriFrasis_2
Input:  Token
Options: control = brill

Macro: DE
(
        {Token.string == "de"}
)

Rule: SintagmaVerbal
Priority: 20
(
  (
    {Token.pos == VA} | {Token.pos == VS} | {Token.pos == VM}
  )
  (
    (DE)
  )?
  (
    {Token.pos == VM}
  )
):pf -->
:pf.SintagmaVerbal = {rule = "SintagmaVerbal",text =:pf@string}
```

Regla 4

```
Phase:  PeriFrasis_3
Input:  Token
Options: control = brill

Rule: SintagmaVerbal
Priority: 20
(
  (
    {Token.pos == VA} | {Token.pos == VS} | {Token.pos == VM}
  )
  (
    {Token.pos == VM}
  )
):pf -->
:pf.SintagmaVerbal = {rule = "SintagmaVerbal",text =:pf@string}
```

Regla 5

```
Phase:  PeriFrasis_4
Input:  Token
Options: control = brill

Macro: PREPOSITION
(
        {Token.string == "por"}
)

Rule: SintagmaVerbal
Priority: 20
(
  (
    {Token.pos == VA} | {Token.pos == VS} | {Token.pos == VM}
  )
  (
    {Token.pos == SP} | (PREPOSITION)
  )?
  (
    {Token.pos == VM}
  )
):pf -->
:pf.SintagmaVerbal = {rule = "SintagmaVerbal",text =:pf@string}
```

Regla 6

```
Phase:  orth
Input:  Token
Options: control = appelt

Macro: VERBOS
(
        {Token.string == "haber"}|
        {Token.string == "ser"}|
        {Token.string == "estar"}
)

Rule: SintagmaVerbal
Priority: 20
(
        (
                {Token.pos == VA}|(VERBOS)
        )?
        (
                {Token.pos == VM}
        )+
):vp -->
:vp.SintagmaVerbal = {rule = "SintagmaVerbal",text =:vp@string}
```

Regla 7

```
Phase:   orth
Input:  Token
Options: control = appelt

Rule: simplePos
(
 {Token}
)
:left
-->
 {
 gate.AnnotationSet toRemove = (gate.AnnotationSet)bindings.get("left");
```

```
 gate.Annotation token = (gate.Annotation)toRemove.iterator().next();

 token.getFeatures().put("gaze","O");
 java.lang.String tempPos = (String)token.getFeatures().get("pos");
 if(tempPos.length()<2)
            token.getFeatures().put("pos",tempPos.substring(0,1));
 else
            token.getFeatures().put("pos",tempPos.substring(0,2));
}
```

Anexo B2. Código para extracción de texto y utilización de reglas.

```
private void cargarReglas() throws ResourceInstantiationException, MalformedURLException {

        // Reglas por default
        reglasJAPE.add((LanguageAnalyser) Factory.createResource("gate.creole.Transducer",
gate.Utils.featureMap("grammarURL", new File(rutaRelativa +
"recursos/reglas/simplifyPosTags.jape").toURI().toURL(), "encoding", "ISO-8859-1")));
        reglasJAPE.add((LanguageAnalyser) Factory.createResource("gate.creole.Transducer",
gate.Utils.featureMap("grammarURL", new File(rutaRelativa +
"recursos/reglas/frasesNominales.jape").toURI().toURL(), "encoding", "ISO-8859-1")));
        reglasJAPE.add((LanguageAnalyser)Factory.createResource( "gate.creole.Transducer",
gate.Utils.featureMap("grammarURL", new File(rutaRelativa +
"recursos/reglas/Perifrasis_0.jape").toURI().toURL(),"encoding", "ISO-8859-1")));
        reglasJAPE.add((LanguageAnalyser)Factory.createResource( "gate.creole.Transducer",
gate.Utils.featureMap("grammarURL", new File(rutaRelativa +
"recursos/reglas/Perifrasis_1.jape").toURI().toURL(),"encoding", "ISO-8859-1")));
        reglasJAPE.add((LanguageAnalyser)Factory.createResource( "gate.creole.Transducer",
gate.Utils.featureMap("grammarURL", new File(rutaRelativa +
"recursos/reglas/Perifrasis_2.jape").toURI().toURL(),"encoding", "ISO-8859-1")));
        reglasJAPE.add((LanguageAnalyser)Factory.createResource( "gate.creole.Transducer",
gate.Utils.featureMap("grammarURL", new File(rutaRelativa +
"recursos/reglas/Perifrasis_3.jape").toURI().toURL(),"encoding", "ISO-8859-1")));
        reglasJAPE.add((LanguageAnalyser) Factory.createResource("gate.creole.Transducer",
gate.Utils.featureMap("grammarURL", new File(rutaRelativa +
"recursos/reglas/Perifrasis_4.jape").toURI().toURL(), "encoding", "ISO-8859-1")));
        reglasJAPE.add((LanguageAnalyser) Factory.createResource("gate.creole.Transducer",
gate.Utils.featureMap("grammarURL", new File(rutaRelativa +
"recursos/reglas/Perifrasis_5.jape").toURI().toURL(), "encoding", "ISO-8859-1")));
        //if(espacio){}
        //if(tiempo){}
        if (causa) {
        }
        if (discursiva) {
        }
    }
```

Anexo C1. Métodos necesarios para cargar plugins de procesamiento del lenguaje español.

```
public RecursoLenguaje extraerInformacionSemantica(RecursoLenguaje corpus) throws InterruptedException {

        try {

            // Registro del plugins
            Gate.getCreoleRegister().registerDirectories(new File(rutaRelativa +
"recursos/gate/plugins/SpanishPlugin").toURI().toURL());
```

```
        Gate.getCreoleRegister().registerDirectories(new File(rutaRelativa +
"recursos/gate/plugins/Tools").toURI().toURL());
        Gate.getCreoleRegister().registerDirectories(new File(rutaRelativa +
"recursos/gate/plugins/ANNIE").toURI().toURL());

        //Agregar documentos al corpus
        Corpus myCorpus = Factory.newCorpus("Textos");
        int x = 0;

        for (String documento : corpus.getTextos()) {
            doc.add(Factory.newDocument(new File(documento).toURI().toURL(), "UTF-8"));

            myCorpus.add(doc.get(x));
            x++;
        }
        //Procesar el corpus
        String[] processingResources = {"vmp.gate.spanish.SpanishSentenceSplitter",
            "vmp.gate.spanish.SpanishTokenizer",
            "vmp.gate.spanish.SpanishPOSTagger",};

        //Agrenado las reglas JAPE correspondientes
        cargarReglas();

        runProcessingResources(processingResources, myCorpus, reglasJAPE);
    } catch (GateException | MalformedURLException e) {
        //e.printStackTrace();
    }
    return corpus;
}
```

Anexo C2. Método que regresa el texto etiquetado en una lista para su representación.

```
public Map<Integer, String> getInformacion(RecursoLenguaje corpus, String etiqueta) {
    String cadenaLimpia;
    String startEnd;
    Map<Integer, String> frasesConId = new TreeMap<>();

    for (int y = 0; y < corpus.getTextos().size(); y++) {
        AnnotationSetImpl ann = (AnnotationSetImpl) doc.get(y).getAnnotations();
        Iterator<Annotation> i = ann.iterator();

        while (i.hasNext()) {
            Annotation annotation = i.next();
            String annotType = annotation.getType();
            if (annotType.contentEquals(etiqueta)) {
                // Se preocesa la cadena para que se eliminen ó reemplazen los caracteres que no acepta OWL
                cadenaLimpia = annotation.getFeatures().get("text").toString();
                if (cadenaLimpia.endsWith(" ")) {
                    cadenaLimpia = cadenaLimpia.substring(0, cadenaLimpia.length() - 1);
                }
                if (cadenaLimpia.startsWith(" ")) {
                    cadenaLimpia = cadenaLimpia.substring(1, cadenaLimpia.length());
                }
                cadenaLimpia = cadenaLimpia.replaceAll("\"\"", "");
                cadenaLimpia = cadenaLimpia.replaceAll("[^A-Za-z0-9áéíóúñ]", "_");          // reemplazar todo lo que
no acepta OWL
                Pattern p = Pattern.compile("^[^A-Za-z]");                                  // indicará si la cadena
inicia con letras
```

```
                Matcher cp2 = p.matcher(cadenaLimpia);                                  // coincidencias del
patron 1
                if (cp2.find()) // No inicia con letras o numeros
                {
                    cadenaLimpia = "_" + cadenaLimpia;
                }

                //Se extrae el offset del texto etiquetado
                startEnd = annotation.getEndNode().getOffset().toString();

                // Se agrega la cadena procesada y el id del texto etiquetado

                frasesConId.put(Integer.parseInt(startEnd), cadenaLimpia);
            }
        }
    }
    return frasesConId;
  }
```

Anexo D1. Clase para la creación y población de la ontología.

```
package edu.conet.pln.ontologia;

import edu.conet.pln.modelo.Tripleta;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.io.OWLXMLOntologyFormat;
import org.semanticweb.owlapi.model.AddAxiom;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLAxiom;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLDataFactory;
import org.semanticweb.owlapi.model.OWLIndividual;
import org.semanticweb.owlapi.model.OWLObjectProperty;
import org.semanticweb.owlapi.model.OWLObjectPropertyAssertionAxiom;
import org.semanticweb.owlapi.model.OWLObjectPropertyDomainAxiom;
import org.semanticweb.owlapi.model.OWLObjectPropertyRangeAxiom;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.model.OWLOntologyStorageException;

import uk.ac.manchester.cs.owl.owlapi.OWLSubClassOfAxiomImpl;

public class Ontologia {

    private static final OWLOntologyManager manejador = OWLManager.createOWLOntologyManager();
    private static final ArrayList<Tripleta> tripleta = new ArrayList<>();
    private static OWLDataFactory factory;
    private static OWLOntology ontologia = null;
    private static OWLObjectProperty opRealizadoPor;
    private static final File output1 = new File("C:/recursos/onto/noticias.rdf"); //Archivo de salida *.rdf
    private static final File output2 = new File("C:/recursos/onto/noticias.owl"); //Archivo de salida *.owl
```

```
    private static final IRI ontologyIRI = IRI.create("http://www.aisii.azc.uam.mx/noticias");//IRI = Dirrección donde estaran las ontologías

    public static ArrayList<Tripleta> creaOntologia(Map<Integer, String> VP, Map<Integer, String> NP) throws OWLOntologyCreationException, OWLOntologyStorageException, IOException {

        ontologia = manejador.createOntology(ontologyIRI);
        factory = manejador.getOWLDataFactory();

        //Se crean las clases de la ontología
        OWLClass claseAgente = factory.getOWLClass(IRI.create(ontologyIRI + "#Agente"));
        OWLClass claseBeneficiario = factory.getOWLClass(IRI.create(ontologyIRI + "#Beneficiario"));
        OWLClass claseCausante = factory.getOWLClass(IRI.create(ontologyIRI + "#Causante"));
        OWLClass claseEfecto = factory.getOWLClass(IRI.create(ontologyIRI + "#Efecto"));
        OWLClass claseEvento = factory.getOWLClass(IRI.create(ontologyIRI + "#Evento"));
        OWLClass claseInstrumento = factory.getOWLClass(IRI.create(ontologyIRI + "#Instrumento"));
        OWLClass claseObjeto = factory.getOWLClass(IRI.create(ontologyIRI + "#Objeto"));
        OWLClass claseUnidadTemporal = factory.getOWLClass(IRI.create(ontologyIRI + "#Unidad_Temporal"));

        //Subclases de la ontolog�a
        OWLClass subclaseEventoNominal = factory.getOWLClass(IRI.create(ontologyIRI + "#Evento_Nominal"));
        OWLClass subclaseEventoVerbal = factory.getOWLClass(IRI.create(ontologyIRI + "#Evento_Verbal"));
        OWLClass subclaseEventoVerbalAfirmativo = factory.getOWLClass(IRI.create(ontologyIRI + "#Evento_Nominal_Afirmativo"));
        OWLClass subclaseEventoVerbalNegativo = factory.getOWLClass(IRI.create(ontologyIRI + "#Evento_Nominal_Negativo"));

        //Se crean los ObjectProperty para implementar la relación entre individuos
        opRealizadoPor = factory.getOWLObjectProperty(IRI.create(ontologyIRI + "#realizado_por"));

        //Agregar clases a la Super clase
        OWLAxiom addAxSupClaseEvn = factory.getOWLDeclarationAxiom(claseEvento);
        OWLAxiom addAxSupClaseAge = factory.getOWLDeclarationAxiom(claseAgente);
        OWLAxiom addAxSupClaseBen = factory.getOWLDeclarationAxiom(claseBeneficiario);
        OWLAxiom addAxSupClaseCau = factory.getOWLDeclarationAxiom(claseCausante);
        OWLAxiom addAxSupClaseEfc = factory.getOWLDeclarationAxiom(claseEfecto);
        OWLAxiom addAxSupClaseIns = factory.getOWLDeclarationAxiom(claseInstrumento);
        OWLAxiom addAxSupClaseObj = factory.getOWLDeclarationAxiom(claseObjeto);
        OWLAxiom addAxSupClaseUnT = factory.getOWLDeclarationAxiom(claseUnidadTemporal);

        //Agregar Subclases a la clase Evento
        OWLSubClassOfAxiomImpl axiomEventoNominal = (OWLSubClassOfAxiomImpl) factory.getOWLSubClassOfAxiom(subclaseEventoNominal, claseEvento);
        OWLSubClassOfAxiomImpl axiomEventoVerbal = (OWLSubClassOfAxiomImpl) factory.getOWLSubClassOfAxiom(subclaseEventoVerbal, claseEvento);

        //Agregar subclases a la clases Evento Verbal
        OWLSubClassOfAxiomImpl axEventoVerbalAfi = (OWLSubClassOfAxiomImpl) factory.getOWLSubClassOfAxiom(subclaseEventoVerbalAfirmativo, subclaseEventoVerbal);
        OWLSubClassOfAxiomImpl axEventoVerbalNeg = (OWLSubClassOfAxiomImpl) factory.getOWLSubClassOfAxiom(subclaseEventoVerbalNegativo, subclaseEventoVerbal);

        //Agregar los axiomas para que dependan de la clase evento
        AddAxiom addAxiomEventoNominal = new AddAxiom(ontologia, axiomEventoNominal);
        AddAxiom addAxiomEventoVerbal = new AddAxiom(ontologia, axiomEventoVerbal);
        AddAxiom addAxEventoVerbalAfi = new AddAxiom(ontologia, axEventoVerbalAfi);
        AddAxiom addAxEventoVerbalNeg = new AddAxiom(ontologia, axEventoVerbalNeg);

        //Se crea la relacion entre individuos Rango
        OWLObjectPropertyRangeAxiom axRangoEve = factory.getOWLObjectPropertyRangeAxiom(opRealizadoPor, claseAgente);
```

```
        //Se crea la relacion entre individuos Dominio
        OWLObjectPropertyDomainAxiom axDominioEve =
factory.getOWLObjectPropertyDomainAxiom(opRealizadoPor, claseEvento);

        //Agregar Axiomas
        AddAxiom addAxiomRango = new AddAxiom(ontologia, axRangoEve);
        AddAxiom addAxiomDominio = new AddAxiom(ontologia, axDominioEve);

        //Ya que se tienen las clases ahora las poblamos con individuos que es extrageron de los textos
        agregarIndividuos(NP, claseAgente, VP, subclaseEventoVerbal);  //NP = NounPhrase para la clase Agente
VP = VerbalPhrase para el Evento Verbal

        //Ahora se van aplicar los cambios con el manejador
        manejador.applyChange(addAxiomRango);
        manejador.applyChange(addAxiomDominio);
        manejador.applyChange(addAxiomEventoNominal);
        manejador.applyChange(addAxiomEventoVerbal);
        manejador.applyChange(addAxEventoVerbalAfi);
        manejador.applyChange(addAxEventoVerbalNeg);
        manejador.addAxiom(ontologia, addAxSupClaseEvn);
        manejador.addAxiom(ontologia, addAxSupClaseAge);
        manejador.addAxiom(ontologia, addAxSupClaseBen);
        manejador.addAxiom(ontologia, addAxSupClaseCau);
        manejador.addAxiom(ontologia, addAxSupClaseEfc);
        manejador.addAxiom(ontologia, addAxSupClaseIns);
        manejador.addAxiom(ontologia, addAxSupClaseObj);
        manejador.addAxiom(ontologia, addAxSupClaseUnT);

        //Aqui se guarda la Ontolog?a en el formato owl
        manejador.saveOntology(ontologia, IRI.create(output1.toURI()));
        manejador.saveOntology(ontologia, new OWLXMLOntologyFormat(), IRI.create(output2.toURI()));
        return tripleta;
    }

    public static void agregarIndividuos(Map<Integer, String> individuo1, OWLClass claseAgente, Map<Integer,
String> individuo2, OWLClass subclaseEventoVerbal) {
        OWLIndividual individual1 = null;
        OWLIndividual individual2 = null;
        Integer i;
        Integer Temp;
        Integer iNumEvento = 0;

        Iterator<Integer> productos1 = individuo1.keySet().iterator();
        Iterator<Integer> productos2 = individuo2.keySet().iterator();
        Iterator<Integer> productos3;
        while (productos1.hasNext() || productos2.hasNext()) {
            Temp = 0;
            if (productos1.hasNext()) {
                i = productos1.next();
                individual1 = factory.getOWLNamedIndividual(IRI.create(ontologyIRI + "#" +
individuo1.get(i)));//Parametro del texto
                manejador.applyChange(new AddAxiom(ontologia, factory.getOWLClassAssertionAxiom(claseAgente,
individual1)));
            }
            if (productos2.hasNext()) {
                i = productos2.next();
                individual2 = factory.getOWLNamedIndividual(IRI.create(ontologyIRI + "#" +
individuo2.get(i)));//Parametro del texto
                manejador.applyChange(new AddAxiom(ontologia,
factory.getOWLClassAssertionAxiom(subclaseEventoVerbal, individual2)));
                iNumEvento = i;
            }
```

```
            //Ciclo que permite obtener el agente más cercano al evento verbal
            productos3 = individuo1.keySet().iterator();
            while (productos3.hasNext() & productos2.hasNext()) {
                i = productos3.next();
                if (i <= iNumEvento & i >= Temp) {
                    Temp = i;
                }
            }
            if (Temp != 0) {
                tripleta.add(new Tripleta(individuo1.get(Temp),"realizado por", individuo2.get(iNumEvento)));
                OWLObjectPropertyAssertionAxiom opAssertinAx =
factory.getOWLObjectPropertyAssertionAxiom(opRealizadoPor, individual2, individual1);
                AddAxiom addAxiomIndividuo = new AddAxiom(ontologia, opAssertinAx);
                manejador.applyChange(addAxiomIndividuo);
            }
        }
    }
}
```

Anexo D2. JSP de Página principal

```
<%--
    Document   : index
    Created on : 11/07/2015, 10:03:41 PM
    Author     : Carlitos
--%>

<%@page import="java.util.List"%>
<%@page import="edu.conet.pln.modelo.Tripleta"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <%@include file='includes/header.jsp' %>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="shortcut icon" href="gumby_mainlogo.png" type="image/x-icon" />
        <title>Poblado de Ontologías</title>
    </head>
    <body>
        <%
            List<Tripleta> tripleta = (List) session.getAttribute("tripleta");
            String nombreDelTexto = (String) session.getAttribute("nombreDelTexto");
            String ruta = "C:/recursos/textos/" + "noticias.rdf";
        %>
        <div class="row" id="tooltips">
            <div class="row">
                <h3 align="center">POBLADO AUTOMATICO DE ONTOLOGÍAS</h3>
                <br>
                <article class="eleven columns">
                    <p align="center">
                        Este sistema web nos permite extraer información
                        útil de textos que servirá para poblar ontologías.
                    </p>
                </article>
                <div class="modal" id="modal1">
                    <div class="content">
                        <a class="close switch" gumby-trigger="|#modal1">Salir</a>
                        <div class="row">
                            <div class="ten columns centered text-center">
```

```
                    <h3>Guia</h3>
                    <p>
                        Este manual sirve de ayuda para la instalacion y un correcto uso de este Sistema Web.
                    </p>
                    <br>
                    <h4 align="center">Poblador Automático de Ontologías</h4>
                    <br>
                    <div>
                        <i class="icon-download">
                            <a href="includes/Manual.pdf">
                                <i class="pretty medium oval secondary btn">
                                    <input type="button" value="Descargar Manual">
                                </i>
                            </a>
                        </i>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <div class="modal" id="modal2">
        <div class="content">
            <a class="close switch" gumby-trigger="|#modal2">Salir</a>
            <div class="row">
                <div class="ten columns centered text-center">
                    <h3>Acerca de..</h3>
                    <br>
                    <p align="justify">
                        El Poblador Automatico de Ontologías fue desarrollado como
                        Proyecto de Integración para la carrera de Ingeniería en computación.
                        El sistema fue desarrollado con las siguientes tecnologías:
                    </p>
                    <p align="justify">
                        <i class="icon-tools">  Apache Tomcat</i><br>
                        <i class="icon-tools">  Oracle JDK</i><br>
                        <i class="icon-tools">  OWL-API</i><br>
                        <i class="icon-tools">  Netbeans IDE.</i><br>
                        <i class="icon-tools">  Protégé 5.0 (Beta)</i><br>
                    </p>
                </div>
            </div>
        </div>
    </div>
    <div class="modal" id="modal3">
        <div class="content">
            <a class="close switch" gumby-trigger="|#modal3">Salir</a>
            <div class="row">
                <div class="ten columns centered text-center">
                    <h3>Ontologías.</h3>
                    <br>
                    <p align="justify">
                        Una ontología es una especificación de una conceptualización,
                        esto es, un marco común o una estructura conceptual sistematizada
                        y de consenso no sólo para almacenar la información, sino también
                        para poder buscarla y recuperarla.
                        Una ontología define los términos y las relaciones básicas para la
                        compresión de un área del conocimiento, así como las reglas para
                        poder combinar los términos para definir las extensiones de este
                        tipo de  vocabulario controlado.
                    </p>
                </div>
            </div>
```

```
                </div>
            </div>
            <div class="modal" id="modal4">
                <div class="content">
                    <a class="close switch" gumby-trigger="|#modal4">Salir</a>
                    <div class="row">
                        <div class="ten columns centered text-center">
                            <h3>Información Desarrollador</h3>
                            <br>
                            <p align="left">
                                Título del proyecto: Poblador Automatico de Ontologías.<br>
                                Trimestre: Primavera 2015 <br>
                                Alumno: Carlos Mauricio Pilapanta Herrera<br>
                                Asesor: Dra. Maricela Claudia Bravo Contreras<br>
                                Co-Asesor: Dr. José Alejandro Reyes Ortiz
                            </p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <hr>
        <%
            String resultado = " ";
            if (session.getAttribute("resultado") != null) {
                resultado = (String) session.getAttribute("resultado");
                if (resultado.equals("ok")) {%>
                    <div class="field">
                        <li class="alert primary">Archivo cargado correctamente!!</li>
                    </div>
                <%} else {%>
                    <p class="field danger">
                        <li class="danger alert">El Archivo no se pudo cargar.</li>
                    </p>
                <%}
                }
                session.removeAttribute("resultado");
        %>
        <form action="subirArchivo.jsp" method="post" enctype="multipart/form-data"
name="formProcesaArchivo">
            <table>
                <tr>
                    <td>
                        <div>
                            <i class="icon-attach">
                                <input class="field" accept=".txt, .pdf" name="archivo" type="file" placeholder="Archivo
de texto"/>
                            </i>
                        </div>
                    </td>
                    <td>
                        <div>
                            <i class="icon-publish" >
                                <i class="pretty medium primary btn">
                                    <input type="submit" value="Subir Archivo">
                                </i>
                            </i>
                        </div>
                    </td>
                </tr>
            </table>
        </form>
```

```
<hr>
<form action="extraerInformacion.jsp" method="post" name="formProcesaTexto" >
   <table>
      <tr>
         <td>
            <input type="hidden" name="nombreDelTexto" value="<%= nombreDelTexto %>">
            <div>
               <i class="icon-cog" >
                  <i class="pretty medium primary btn">
                     <input type="submit" value="Procesar Texto">
                  </i>
               </i>
            </div>
         </td>
         <td>
            <strong><h4>DESCARGAR owl</h4></strong>
         </td>
         <td>
            <div>
               <i class="icon-download">
                  <a href="includes/noticias.rdf">
                     <i class="pretty medium primary btn">
                        <input type="button" value="Descargar  owl">
                     </i>
                  </a>
               </i>
            </div>
         </td>
      </tr>
   </table>
</form>
<%
   String resultado1 = " ";
   if (session.getAttribute("resultado1") != null) {
      resultado1 = (String) session.getAttribute("resultado1");
      if (resultado1.equals("ok")) {%>
         <div class="field success">
            <li class="alert primary">Se proceso el texto correctamente!!</li>
         </div>
      <%} else {%>
         <p class="field alert">
            <li class="danger alert">No se pudo procesar el texto.</li>
         </p>
      <%}
     }
     session.removeAttribute("resultado1");
%>
<hr>
<h2 class="medium text-center">Resultados</h2>
<div>
   <table>
      <thead>
         <tr>
            <th>
               Individuo
            </th>
            <th>
               Relación
            </th>
            <th>
               Agente
            </th>
```

```
                    </tr>
                </thead>
                    <%
                        if(tripleta != null){
                            for(int i =0 ;i < tripleta.size();i++){ %>
                                <tr>
                                    <td>
                                        <%= tripleta.get(i).getObjeto().replaceAll("_", " ") %>
                                    </td>
                                    <td>
                                        <%= tripleta.get(i).getPredicado() %>
                                    </td>
                                    <td>
                                        <%= tripleta.get(i).getSujeto().replaceAll("_", " ") %>
                                    </td>
                                </tr>
                        <%  }
                        }else{%>
            </table>
                    <strong><h4 class="medium text-center">No Hay resultados</h4></strong>
                     <% }
                        session.removeAttribute("tripleta");
                     %>
        </div>
    </div>
</body>
<%@include file='includes/scripts.jsp' %>
</html>
```