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

Reporte final del proyecto de Integración:

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

"Proyecto Tecnológico"

Alumno
Josué Padilla Cuevas 210204682

Datos de asesor:
Maricela Claudia Bravo Contreras
Profesor Asociado
Departamento de Sistemas

Datos de co-asesor.
José Alejandro Reyes Ortiz
Profesor Titular
Departamento de Sistemas

Trimestre 2014 Otoño

Fecha de entrega
10 de Diciembre de 2014

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 UAM Azcapotzalco.

Firma del asesor

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 UAM Azcapotzalco.

Firma del asesor

Yo, Josué Padilla Cuevas, 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.

Firma del alumno

RESUMEN

En la actualidad, la información representa un elemento de gran importancia dentro de nuestras vidas. La tecnología moderna nos ha puesto delante volúmenes increíbles de información, mucha de la cual por estar disponible en textos escritos en lenguaje natural sin restricciones, necesita de un procesamiento previo para poder ser usada y aplicada a la resolución de los problemas que tenemos que enfrentar (conocimiento).

La realización de este proyecto surgió para darle una respuesta puntual a la problématica que se tiene con el procesamiento de información disponible en textos periodísticos escritos en lenguaje español. Este proyecto contiene el diseño e implementación de un sistema que realiza la extracción de información en noticias nacionales, mediante reglas lingüísticas y su representación en un modelo de datos semántico, esto facilita el acceso de grandes volúmenes de texto y mejora así la productividad en las tareas de análisis y síntesis de información.

Este proyecto de integración nos permite procesar y clasificar la información encontrada en los textos periodísticos. Con la ayuda de técnicas de inteligencia artificial, específicamente de extracción de información basada en las reglas lingüísticas, permitiendo identificar información relevante, en este caso: el lugar y la fecha en la que se llevaron a cabo los hechos.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

1. INTRODUCCIÓN

En la actualidad, la información constituye un elemento de gran importancia dentro de nuestras vidas. La tecnología moderna nos ha puesto delante volúmenes increíbles de información, mucha de la cual por estar disponible en textos escritos en lenguaje natural sin restricciones, necesita de un procesamiento previo para poder ser usada y aplicada a la resolución de los problemas que tenemos que enfrentar.

La extracción de información es una tarea de procesamiento de lenguaje natural cuyo propósito es extraer determinados tipos de información de un documento. Los sistemas de extracción de información se enfocan en un dominio específico, esto debido a que extraen información relevante, como eventos, hechos, lugares, tiempos, entre otros. Esta tarea no es trivial y por lo tanto requiere de algoritmos computacionales eficientes. En este documento se diseñó e implementó un sistema que realice la extracción de información en textos periodísticos nacionales mediante reglas lingüísticas facilitando el acceso de grandes colecciones de datos textuales y mejorando así la productividad en las tareas de análisis y síntesis de información.

2. ANTECEDENTES

2.1 Referencias Internas

Dentro de la Universidad Autónoma Metropolitana, existen diferentes Proyectos Terminales que abarcan un conjunto de temas relacionados con la ingeniería la extracción de información utilizando procesamiento del lenguaje natural y el almacenamiento de datos en modelos semánticos.

Como es el caso del Proyecto que lleva por nombre “Sistema de recuperación de información semántico [1]”, en este proyecto se llevó a cabo la extracción de información mediante lenguaje natural, compartiendo con este proyecto la idea principal que es: recabar información necesaria, extraerla y posteriormente procesar los resultados.

Otro de los proyectos que se pueden mencionar es el titulado “Sistema clasificador de documentos de proyectos terminales usando el concepto de memoria asociativa [2]” este sistema realiza un procesamiento de archivos de texto para poder manipular la información requerida y así disminuir el tiempo invertido en revisar archivo por archivo.

Finalmente se mencionará el proyecto “Sistema de procesamiento de textos de investigación [3]”, en este se diseñó un sistema de anotado de textos basado en reglas lingüísticas.

2.2 Referencias Externas

Extracción de información con algoritmos de clasificación [4].

En esta tesis se dan las bases teóricas para la extracción de información, nos muestra los algoritmos que se deben de utilizar, como por ejemplo los k-vecinos más cercanos, así como la utilización de autómatas de textos, el proyecto que vamos a implementar se basará en esta tesis, esto con el fin de conocer el aprendizaje de reglas y el aprendizaje estadístico, la diferencia del proyecto que estamos proponiendo es que este será de carácter estrictamente práctico, esto quiere decir que llevaremos a cabo la implementación de módulos para la extracción de información basado en reglas lingüísticas.

Utilizando recursos lingüísticos para la mejora de recuperación de información en la web [5]

En este trabajo de investigación se muestran aspectos relacionados con la integración de la tecnología disponible de tratamiento de lenguaje natural en el desarrollo de un meta buscador que alcance un mayor grado de acierto en la recuperación de información, así como el tratamiento posterior de los archivos obtenidos, para ello se utiliza un recurso lingüístico así como dos recursos léxicos: Aries para el tratamiento de la morfología y EuroWordnet para el tratamiento de la semántica. El proyecto propuesto realizará algo similar solo que estará acotado a textos periodísticos nacionales y utilizaremos otras herramientas para el procesamiento del texto.

ReVerb [6]

Reverb es un programa que automáticamente identifica y extrae relaciones binarias de oraciones en inglés. Está diseñado para la extracción de información de escala Web, donde las relaciones de destino no se pueden especificar con antelación y la velocidad es importante. El proyecto propuesto obtendrá a partir de archivos de texto la información temporal y espacial, la principal diferencia con este software es que este proyecto utilizará textos en idioma español.

Por otro lado durante los últimos años, la utilización de la web semántica ha sido de gran utilidad para consolidar lo que podríamos definir como el uso actual del término ontología. Dentro de este contexto, una de las definiciones mayormente mencionadas es la de *Studer* quien define la ontología como “una especificación explícita y formal de una conceptualización compartida”.

En la actualidad las aplicaciones de las ontologías son múltiples, son utilizadas en gestión de conocimiento, procesamiento de lenguaje natural, comercio electrónico, integración y recuperación inteligente de información, educación y la Web Semántica entre algunas otras.

3. JUSTIFICACIÓN

La clasificación de información encontrada en los textos periodísticos involucra un alto uso de tiempo y esfuerzo; debido a esto se diseñó un sistema de información que permitió facilitar la extracción de datos para poder gestionarla con mayor facilidad por parte de los usuarios reduciendo considerablemente el tiempo.

En este proyecto se llevó a cabo la realización de una aplicación que requiere de técnicas de inteligencia artificial, específicamente de extracción de información basada en las reglas lingüísticas. Al ser un sistema de información basado en extracción de información relevante mediante algoritmos complejos se justifico que fuera un proyecto de integración para la carrera de Ingeniería en Computación; y será integrado en un proyecto de investigación del Departamento de Sistemas en el Grupo de Investigación en Sistemas de Información Inteligentes que se llevará a cabo en la Universidad Autónoma Metropolitana. Además, este proyecto apoyaró la generación de resúmenes de noticias y análisis estadístico de eventos.

4. OBJETIVOS

4.1 Objetivo General

Extraer información espacial y temporal a partir de textos periodísticos descritos en español aplicando reglas lingüísticas con la finalidad de enriquecer un modelo de datos semánticos.

4.2 Objetivos Específicos

- Diseñar e implementar un modelo de datos semánticos para la representación de información espacial y temporal.
- Diseñar e implementar un módulo para el procesamiento de textos periodísticos.
- Implementar un algoritmo basado en reglas lingüísticas para la extracción de información.
- Instanciar la información espacial y temporal en el modelo de datos semántico.
- Diseñar e implementar una aplicación que integre el proceso completo de extracción de información y su visualización en los textos periodísticos.

5. MARCO TEÓRICO

5.1 Procesamiento del lenguaje Natural

Una de las ramas más importantes de la Inteligencia Artificial es aquella orientada a facilitar la comunicación hombre-computadora por medio del lenguaje humano, o lenguaje natural. El Procesamiento del Lenguaje Natural (PLN) es la disciplina encargada de producir sistemas informáticos que posibiliten dicha comunicación, por medio de la voz o del texto. Se trata de una disciplina tan antigua como el uso de las computadoras (años 50), de gran profundidad, y con aplicaciones tan importantes como la traducción automática o la búsqueda de información en Internet. Dado el tiempo disponible, es imperativo concentrar nuestros esfuerzos en un ámbito necesariamente limitado: los sistemas de PLN que utilizan técnicas de carácter estadístico aplicados al análisis del texto [7].

Es una disciplina con una larga trayectoria. Nace en la década de 1960, como área dentro de la taxonomía de la Inteligencia Artificial y la Lingüística, con el objeto de estudiar los problemas derivados de la generación y comprensión automática del lenguaje natural.

En sus orígenes, sus métodos tuvieron gran aceptación y éxito, no obstante, cuando sus aplicaciones fueron llevadas a la práctica, en entornos no controlados y con vocabularios genéricos, empezaron a surgir multitud de dificultades. Entre ellas, pueden mencionarse por ejemplo los problemas de polisemia y sinonimia.

En los últimos años, las aportaciones que se han hecho desde este dominio han mejorado sustancialmente, permitiendo el procesamiento de ingentes cantidades de información en formato texto con un grado de eficacia aceptable. Muestra de ello es la aplicación de estas técnicas como una componente esencial en los motores de búsqueda web, en las herramientas de traducción automática, o en la generación automática de resúmenes.

5.1.1 Problemática del procesamiento del lenguaje natural

El lenguaje natural, entendido como la herramienta que utilizan las personas para expresarse, posee propiedades que merman la efectividad de los sistemas de recuperación de información textual. Estas propiedades son la variación y la ambigüedad lingüística. Cuando hablamos de la variación lingüística nos referimos a la posibilidad de utilizar diferentes palabras o expresiones para comunicar una misma idea. En cambio, la ambigüedad lingüística se produce cuando una palabra o frase permite más de una interpretación.

Ambos fenómenos inciden en el proceso de recuperación de información aunque de forma distinta. La variación lingüística provoca el silencio documental, es decir la omisión de documentos relevantes para cubrir la necesidad de información, ya que no

se han utilizado los mismos términos que aparecen en el documento. En cambio, la ambigüedad implica el ruido documental, es decir la inclusión de documentos que no son significativos, ya que se recuperan también documentos que utilizan el término pero con significado diferente al requerido. Estas dos características dificultan considerablemente el tratamiento automatizado del lenguaje. A continuación se muestran algunos ejemplos que ilustran la repercusión de estos fenómenos en el proceso de recuperación de información:

A nivel morfológico una misma palabra puede adoptar diferentes roles morfo-sintácticos en función del contexto en el que aparece, ocasionando problemas de ambigüedad.

Ejemplo 1. Deja la comida que sobre la mesa de la cocina, dijo llevando el sobre en la mano.
La palabra sobre es ambigua morfológicamente ya que puede ser un sustantivo masculino singular, una preposición, y también la primera o tercera persona del presente de subjuntivo del verbo sobrar.

A nivel sintáctico, centrado en el estudio de las relaciones establecidas entre las palabras para formar unidades superiores, sintagmas y frases, se produce ambigüedad a consecuencia de la posibilidad de asociar a una frase más de una estructura sintáctica. Por otro lado, esta variación supone la posibilidad de expresar lo mismo pero cambiando el orden de la estructura sintáctica de la frase.

Ejemplo 2. María vio a un niño con un telescopio en la ventana.
La interpretación de la dependencia de los dos sintagmas preposicionales, con un telescopio y en la ventana, otorga diferentes significados a la frase: (i) María vio a un niño que estaba en la ventana y que tenía un telescopio, (ii) María estaba en la ventana, desde donde vio a un niño que tenía un telescopio, y (iii) María estaba en la ventana, desde donde miraba con un telescopio, y vio a un niño.
A nivel semántico, donde se estudia el significado de una palabra y el de una frase a partir de los significados de cada una de las palabras que la componen. La ambigüedad se produce porque una palabra puede tener uno o varios sentidos, es el caso conocido como polisemia.

Ejemplo 3. Luís dejó el periódico en el banco.
El término banco puede tener dos significados en esta frase, (i) entidad bancaria y (ii) asiento. La interpretación de esa frase va más allá del análisis de los componentes que forman la frase, se realiza a partir del contexto en que es formulada.
Y también hay que tener en cuenta la variación léxica que hace referencia a la posibilidad de utilizar términos distintos a la hora de representar un mismo significado, es decir el fenómeno conocido como sinonimia.

Ejemplo 4: Coche / Vehículo / Automóvil.
A nivel pragmático, basado en la relación del lenguaje con el contexto en que es utilizado, en muchos casos no puede realizarse una interpretación literal y automatizada de los términos utilizados. En determinadas circunstancias, el sentido de las palabras que forman una frase tiene que interpretarse a un nivel superior recurriendo al contexto en que es formulada la frase.

Ejemplo 5. Se moría de risa.
En esta frase no puede interpretarse literalmente el verbo morirse si no que debe entenderse en un sentido figurado.
Otra cuestión de gran importancia es la ambigüedad provocada por la anáfora, es decir, por la presencia en la oración de pronombres y adverbios que hacen referencia a algo mencionado con anterioridad.

Ejemplo 6. Ella le dijo que los pusiera debajo
La interpretación de esta frase tiene diferentes incógnitas ocasionadas por la utilización de pronombres y adverbio: ¿quién habló?, ¿a quién?, ¿qué pusiera qué?, ¿debajo de dónde? Por tanto, para otorgar un significado a esta frase debe recurrirse nuevamente al contexto en que es formulada.
Con todos los ejemplos expuestos queda patente la complejidad del lenguaje y que su tratamiento automático no resulta fácil ni obvio.

Ejemplo 7 El auto come niños.
Es sintácticamente correcta pero semánticamente incorrecta.
El nivel de análisis semántico utiliza los rasgos semánticos de las palabras.

5.1.2 PLN en la recuperación de información textual

La complejidad asociada al lenguaje natural cobra especial relevancia cuando necesitamos recuperar información textual que satisfaga la necesidad de información de un usuario. Es por ello, que en el área de Recuperación de Información Textual las técnicas de NLP son muy utilizadas, tanto para facilitar la descripción del contenido de los documentos, como para representar la consulta formulada por el usuario, y ello, con el objetivo de comparar ambas descripciones y presentar al usuario aquellos documentos que satisfagan en mayor grado su necesidad de información.

Un sistema de recuperación de información textual lleva a cabo las siguientes tareas para responder a las consultas de un usuario:

1- Indexación de la colección de documentos: en esta fase, mediante la aplicación de técnicas de NLP, se genera un índice que contiene las descripciones de los

documentos. Normalmente, cada documento es descrito mediante el conjunto de términos que, hipotéticamente, mejor representa su contenido.

2- Cuando un usuario formula una consulta el sistema la analiza, y si es necesario la transforma, con el fin de representar la necesidad de información del usuario del mismo modo que el contenido de los documentos.
3- El sistema compara la descripción de cada documento con la descripción de la consulta, y presenta al usuario aquellos documentos cuyas descripciones más se asemejan a la descripción de su consulta.
4- Los resultados suelen ser mostrados en función de su relevancia, es decir, ordenados en función del grado de similitud entre las descripciones de los documentos y de la consulta.

Estas tareas de extracción de información textual las podemos ver representadas en el diagrama de flujo de la figura 1.

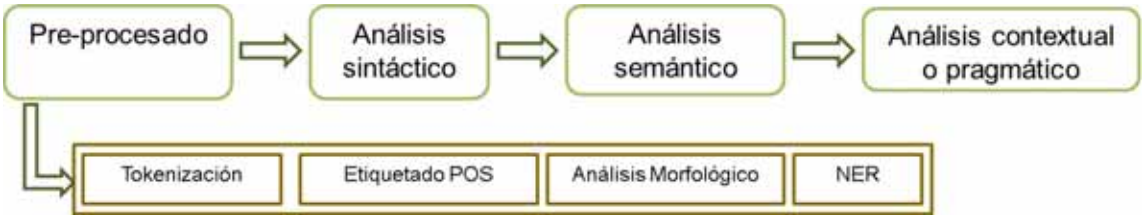

Figura 1. Diagrama de extracción de información [9]

De momento no existen técnicas de NLP que permitan extraer de forma inequívoca el significado de un documento o una consulta. De hecho, la comunidad científica está dividida en cuanto a los procedimientos a seguir para alcanzar tal objetivo.

5.1.3 Procesamiento lingüístico del lenguaje natural

Esta aproximación se basa en la aplicación de diferentes técnicas y reglas que codifican de forma explícita el conocimiento lingüístico. Los documentos son analizados a partir de los diferentes niveles lingüísticos, citados ya anteriormente, por herramientas lingüísticas que incorporan al texto las anotaciones propias de cada nivel. A continuación se muestran los diferentes pasos a realizar para llevar a cabo un análisis lingüístico de los documentos aunque ello no implica que se apliquen en todos los sistemas.

El análisis morfológico es ejecutado por los etiquetadores (taggers) que asignan a cada palabra su categoría gramatical a partir de los rasgos morfológicos identificados.

Después de identificar y analizar las palabras que forman un texto, el siguiente paso consiste en ver cómo éstas se relacionan y combinan entre sí para formar unidades

superiores, los sintagmas y las frases. Por tanto, se trata de realizar el análisis sintáctico del texto. En este punto se aplican gramáticas (parsers) que son formalismos descriptivos del lenguaje que tienen por objetivo fijar la estructura sintáctica del texto. Las técnicas empleadas para aplicar y construir las gramáticas son muy variadas y dependen del objetivo con el que se realiza el análisis sintáctico. En el caso de la recuperación de la información acostumbra a aplicarse un análisis superficial, donde se identifican únicamente las estructuras más significativas: frases nominales, sintagmas verbales y preposicionales, entidades, etc. Este nivel de análisis suele utilizarse para optimizar recursos y no ralentizar el tiempo de respuesta de los sistemas.

A partir de la estructura sintáctica del texto, el siguiente objetivo es obtener el significado de las frases que lo componen. Se trata de conseguir la representación semántica de las frases, a partir de los elementos que la forman.

Una de las herramientas más utilizadas en el procesamiento semántico es la base de datos lexicográfica WordNet. Se trata de un léxico semántico anotado en diferentes lenguas, formado por grupos de sinónimos llamados synsets de los que se facilitan definiciones cortas y se almacenan las distintas relaciones semánticas entre estos grupos de sinónimos [8].

5.1.4 Identificación de estructuras

Se trata de encontrar, dado un texto, informaciones muy concretas que suelen adoptar estructuras similares. Esto permite emplear patrones que combinan información de estructura con información lingüística.

- ✓ Números de teléfono: (NNN) NNN-NN NN
- ✓ Direcciones postales : C.P. NNNNN
- ✓ Direcciones de correo electrónico.

5.1.5 Identificación de palabras clave

Determinar de forma automática qué palabras de un texto resultan más adecuadas para caracterizarlo, es decir, qué palabras deben elegirse como posibles palabras clave.

La correcta combinación de la frecuencia de aparición de una palabra en el texto junto con su frecuencia global.

5.1.6 Reconocimiento de entidades con nombre

La posibilidad de reconocer automáticamente la aparición de un nombre propio en un texto es una de las aplicaciones más útiles de la extracción de información. Distinguir cuándo se habla de una persona, una organización, un lugar, etc.

5.2 Modelo de datos semántico

5.2.1 Ontología

A lo largo de la existencia del hombre, el conocimiento humano ha tenido la necesidad de ser compartido entre las personas, es por ello que se desarrollaron diferentes medios para tal fin, como imágenes, lenguaje hablado y lenguaje escrito.

Sin embargo, en los últimos años surge la necesidad de compartir conocimiento entre maquinas considerando cuidadosamente la forma de almacenarlo para que sea legible por las computadoras y reutilizable. Esta necesidad surge para facilitar las tareas de un dominio (por ejemplo compartir información, realizar deducciones, recuperar información). Por lo tanto, la representación del dominio debe expresar con claridad los hechos, esto introduce un concepto proveniente de las filosofías y utilizado en Inteligencia Artificial: las ontologías.

El término ontología proviene de los Griegos y significa 'hablar' (-logía) acerca de la 'existencia' (onto-). Sin embargo, la palabra 'Ontología' es una disciplina psicológica que puede ser descrita como la ciencia de la existencia o el estudio del ser. En el área de las ciencias computacionales, las ontologías aparecen en la década de los 80's como un nuevo modelo para compartir y reutilizar el conocimiento, fue hasta mediados de los 90's que empiezan a aplicarse a la web en la publicación de descripciones semánticas explícitas de recursos (contenidos, servicios, imágenes, entre otros). En la actualidad son el eje fundamental en las nuevas tecnologías para la Web semántica.

La importancia que han alcanzado las ontologías se debe en gran medida a que constituyen una herramienta poderosa para varias tareas como: procesamiento de lenguaje natural, filtrado de información, recuperación de información, acceso a datos [9].

Existen varias definiciones de ontología, a continuación se incluyen propuestas concretas en el área de las ciencias computacionales.

"Taxonomía de conceptos con atributos y relaciones, que proporciona un vocabulario consensuado para definir redes semánticas de unidades de información interrelacionadas [10]".

Una especificación explicita y formal sobre una conceptualización compartida [10]".

Una ontología es una base de datos donde se describen conceptos del mundo o algún dominio en específico, sus propiedades y como se relacionan los conceptos entre si [11]".

La interpretación de estas definiciones es que las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada, donde esta conceptualización debe ser representada de manera formal, legible y compartida por maquinas. La clasificación de las ontologías con respecto a su nivel de dependencia se puede observar en la Figura 2 y es descrita a continuación:

- **Ontologías de un dominio**, representan el conocimiento especializado de un dominio o subdominio y son independiente de la aplicación.

- **Ontologías de nivel superior**, describen conceptos generales y fundacionales del conocimiento como las estructuras parte-todo, la cuantificación, los procesos o los tipos de objetos.

- **Ontologías de aplicación**, contienen todas las definiciones necesarias para modelar el conocimiento de una determinada aplicación, y generalmente son una especialización de ontologías de dominio.

- **Ontologías de la tarea**, proporcionan un vocabulario sistemático de los términos utilizados para resolver problemas relacionados con las tareas que pueden o no pertenecer al mismo dominio.

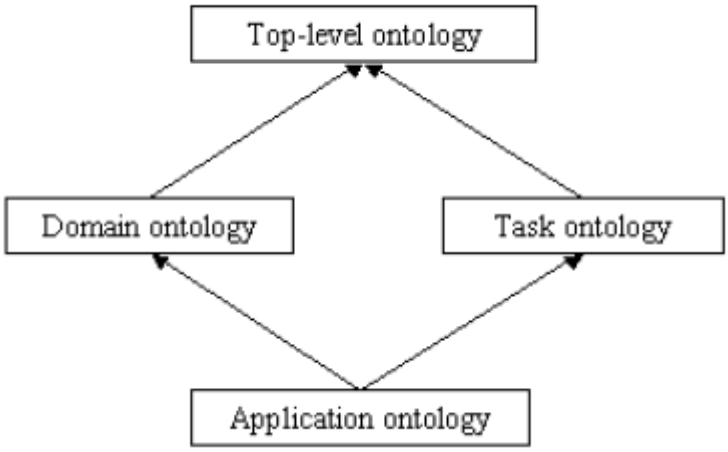

Figura 2. Clasificación de las ontologías.

5.2.2 Ontologías para el Procesamiento del Lenguaje Natural y la Terminología

En el ámbito del Procesamiento del Lenguaje Natural, las ontologías se están empleando para construir representaciones independientes de la lengua que puedan servir de punto de encuentro entre dos o más lenguas naturales. En este sentido la ontología se considera como el repositorio de conceptos que establecen conexiones entre los símbolos de una lengua y sus referentes en el mundo o submundo que se contempla.

6. DESARROLLO DEL PROYECTO

A continuación se va a explicar el desarrollo del sistema, los módulos que fueron implementados están esquematizados en la Figura 3.

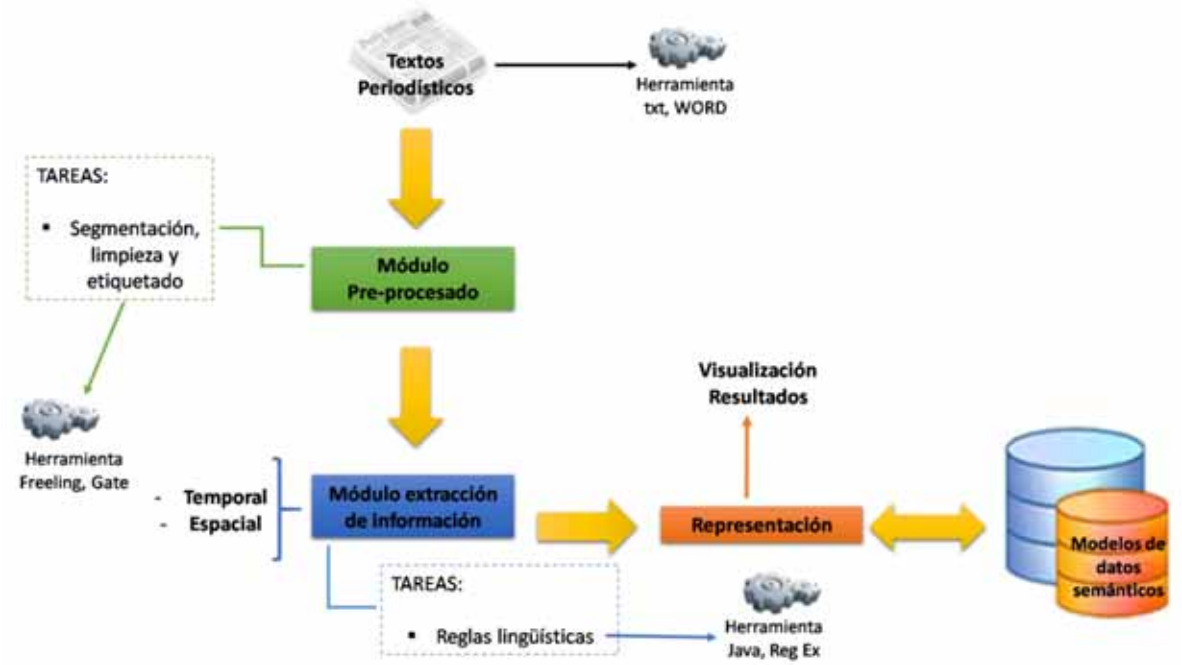

Figura 3. Arquitectura del sistema para la extracción de información espacial y temporal en textos periodísticos mediante reglas lingüísticas.

6.1 Elaboración del módulo de pre-procesado

Esta es la primera etapa que se desarrolló del sistema, se compone de tres fases, la limpieza de las noticias, la segmentación y finalmente el etiquetado.

6.1.1 Limpieza de las noticias

Las noticias obtenidas de un corpus se envían al módulo de limpieza, este módulo se encarga de buscar dentro de un listado de palabras vacías[1] todas las coincidencias existentes dentro de las noticias, al encontrar alguna de estas palabras que no aporta significado relevante simplemente las sustituye en la noticia por un espacio en blanco. Las palabras vacías utilizadas en esta fase se muestran en la Figura 4. Una vez que el módulo de limpieza quita todas las palabras sin significado que encuentre dentro del texto, da como salida la noticia limpia, en la Figura 5 podemos ver un ejemplo de lo que realiza este módulo.

[1] Palabras vacías: nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc.

acuerdo,adelante,ademas,además,adrede,ahi,ahí,ahora,al,alli,allí,alrededo
r,antano,antaño,antes,apenas,aproximadamente,aquel,aquél,aquella,aquélla,
aquellas,aquéllas,aquello,aquellos,aquéllos,aqui,aquí,arriba,asi,así,aun,
aún,aunque,bajo,bastante,bien,breve,casi,cerca,claro,como,cómo,con,conmig
o,contigo,contra,cual,cuál,cuales,cuáles,cuando,cuándo,cuanta,cuánta,cuan
tas,cuántas,cuanto,cuánto,cuantos,cuántos,debajo,delante,demasiado,dentro
,deprisa,despacio,despues,después,detras,detrás,dia,día,dias,días,donde,d
ónde,dos,durante,él,ella,ellas,ellos,encima,enfrente,enseguida,entre,es,e
sa,ésa,esas,ésas,ese,ése,eso,esos,ésos,está,ésta,estado,estados,estan,est
án,estar,estas,éstas,éste,esto,estos,éstos,ex,excepto,final,fue,fuera,fue
ron,g,general,gran,ha,habia,había,habla,hablan,hace,hacia,han,hasta,hay,h
oras,hoy,incluso,informo,informó,junto,lado,las,le,lejos,lo,los,luego,mal
,mas,más,mayor,me,medio,mejor,menos,menudo,mi,mí,mia,mía,mias,mías,mientr
as,mio,mío,mios,míos,mis,mismo,mucho,muy,n,nada,nadie,ninguna,no,nos,noso
tras,nosotros,nuestra,nuestras,nuestro,nuestros,nueva,nuevo,nunca,os,otra
,otros,pais,país,parte,pasado,peor,pero,poco,p_or,porque,pronto,proximo,p
róximo,puede,que,qué,quien,quién,quienes,quiénes,quiza,quizá,quizas,quizá
s,raras,repente,salvo,se,sé,segun,según,ser,sera,será,si,sí,sido,siempre,
sin,so_bre,solamente,solo,sólo,son,soyos,su,supuesto,sus,suya,suyas,suyo,
t,tal,tambien,también,tampoco,tarde,te,temprano,ti,tiene,todavia,todavía,
todo,todos,tras,tu,tú,tus,tuya,tuyas,tuyo,tuyos,u,un,una,unas,uno,unos,us
ted,ustedes,v,veces,vez,vosotras,vosotros,vuestra,vuestras,vuestro,vuestr
os,w,x,y,ya,yo,z

Figura 4. Palabras vacías

Figura 5. Módulo de limpieza

6.1.2 Segmentación de las noticias.

Esta parte del módulo se desarrolló con la ayuda de la herramienta TreeTagger [12], al enviarle una cadena de texto al módulo, está la parte en tokens o viéndolo de un modo más simple guarda cada palabra en una estructura de datos, esto es quitar los espacios dentro de la noticia y segmentarla por palabras, la salida de este módulo es un arreglo de cadenas donde se almacenan todas las palabras que componen la noticia.

La actividad que realiza este módulo se ve representada en el diagrama de la Figura 6.

Figura 6. Módulo de segmentación

6.1.3 Etiquetado de las noticias

La parte final del módulo de pre-procesado es el etiquetado, este se desarrolló de la siguiente manera:

Una vez que se recibe del módulo anterior la noticia segmentada en palabras, se utiliza la herramienta TreeTagger[2] ya que gracias a sus métodos podemos realizar un etiquetado para determinar las dependencias gramaticales y para hacer un análisis léxico, esto es, clasificar las palabras del texto en categoría: palabra, número, puntuación, etc. Una vez etiquetadas las palabras estas se guardan en una estructura de datos bidimensional, donde tendremos el lema y el pos (clasificación de las palabras). La salida de este módulo serán varias cadenas de palabras clave pre-procesadas y etiquetadas gramaticalmente (ejemplo Figura 7).

Figura 7. Módulo de etiquetado

[2] TreeTagger es una herramienta para la anotación de texto con la parte de expresión y de información lema.

En la figura 8 podemos ver la aplicación de escritorio TreeTagger

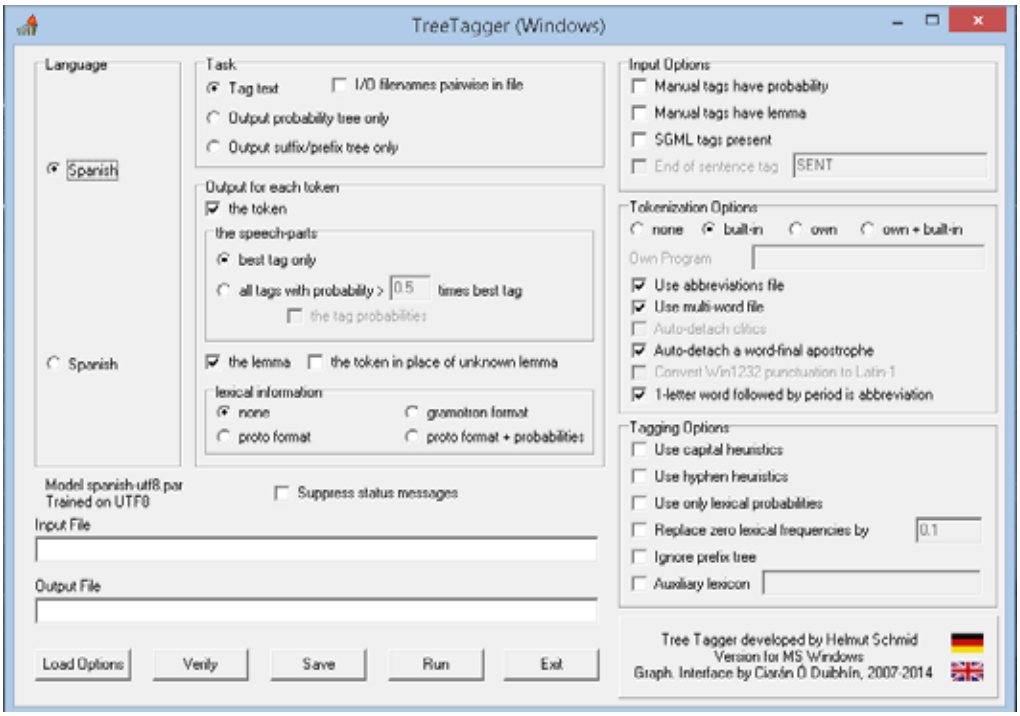

Figura 8. Aplicación de escritorio TreeTagger [12]

La implementación de este módulo se realizó en Java y su código se puede consultar en el Anexo A.

6.2 Elaboración del módulo de extracción de información

Este módulo es el encargado de extraer la información temporal y espacial del texto periodístico, para llevar a cabo esta tarea se implementó en java mediante expresiones Regulares (API de RegEx[3]) basadas en técnicas de Procesamiento de Lenguaje Natural, específicamente reglas lingüísticas.

El módulo de extracción recibe de entrada las palabras procesadas de los textos periodísticos nacionales obtenidos del módulo anterior, se utilizan métodos de procesamiento de lenguaje natural en concreto reglas lingüísticas para poder encontrar la información temporal y espacial en dichos textos. Su salida son las palabras o frases donde tengamos referencia de información temporal (cuándo ocurrió el evento) y espacial (dónde ocurrió el evento) que son almacenadas en un modelo de datos semántico para su gestión.

[3] Una expresión regular, a menudo llamada también regex, es una secuencia de caracteres que forma un patrón de búsqueda.

6.2.1 Extracción de información espacial en los textos periodísticos

Para llevar a cabo el desarrollo de esta parte del módulo, fue necesario implementar reglas lingüistas, con el objetivo de encontrar una estructura que nos pudiera referenciar donde ocurrieron los hechos en la noticia.

Las entidades de espacio a identificar se basan en la clasificación mostrada en la Tabla 1. La extracción se basa en buscar e identificar un tipo de marcador lingüístico: urbano o rural. Este marcador debe estar seguido de un nombre propio. En la Tabla 1, se muestra las clases del espacio urbano y espacio natural, asi como sus sinónimos.

La extracción de estas entidades espaciales se realiza con la aplicación de los siguientes patrones:

Patrones

P1 → (en) NP

P2 → (en) *{1} NP (donde * puede valer una o dos palabras)

P2→ (en) *{1-2} NP

P3→ (en|por|sobre) [MarcadorLingUrbano] * (NP or NC)
MarcadorLingUrbano no tiene que ser NP porque sería un apellido. Ej. Del Bosque

P4→ (en|por|sobre) [MarcadorLingNatural] * (NP or NC)

P5 → [MarcadorLingUrbano] *{1-2} (NP or NC)

P6 → [MarcadorLingNatural] *{1-2} (NP or NC)

La implementación de estos patrones se realizaron en Java y su código se puede consultar en el Anexo B.
El resultado de la extracción después de aplicar los patrones de información espacial a un texto periodístico, se muestran en la figura 9. Este resultado contesta la pregunta *¿Dónde ocurrieron los hechos de la notica?* En este ejemplo se aplica el patrón P1 para la extracción de información. La información extraida se utiliza para crear las instancias de la ontología de espacio.

> Este activista ucraniano muy activo dentro de la oposición contra el presidente ViktorYanukovitch afirma haber sido secuestrado el 22 de enero en Kiev y torturado durante una semana, antes de ser liberado en medio de un bosque.

Figura 9. Resultado extracción espacial.

Clase	Tipo de espacio	Sinónimos
Espacio Urbano	Plaza	
	Ciudad	Metrópolis, Centro urbano, capital
	Escuela	colegio, academia, instituto, liceo, conservatorio, facultad
	Edificio	inmueble, casa, vivienda, obra, bloque, finca, construcción, edificación, fábrica, manzana
	Localidad	Municipio, población, aldea, pueblo, villa
	Colonia	Barrio, condominio
	Mercado	feria, mercadillo, bazar, supermercado
	Estado	jurisdicción, provincia, zona
	País	Nación, territorio,
	Calle	vía, paseo, avenida, boulevard, ronda, carrera, callejón, travesía, pasadizo, pasaje, calzada
Espacio Natural	Bosque	selva, espesura, frondosidad, boscaje, arbolado, huerto
	Montaña	cordillera, cadena, monte, cumbre, cima, prominencia, promontorio, altura, elevación, cerro, colina, montículo, loma
	Sierra	Cumbre, pico, serranía
	Lago	Albufera, estanque, laguna, marisma, alberca, pantano, balsa, charca, estero.
	Rio	torrente, arroyo, torrentera, riachuelo, riacho, afluente, regato, abundancia, caudal, profusión, raudal, cantidad, plétora, barranco
	Parque	Jardín, floresta, edén, alameda, arboleda, reserva natural.
	Acantilado	Rápido, abismo, salto, precipicio

Tabla 1. Espacio urbano y natural

6.2.2 Extracción de información temporal en los textos periodísticos

Para el desarrollo de la segunda parte de este módulo, fue necesaria la implementación de patrones de reglas lingüistas, con el objetivo de encontrar una estructura de tiempo (fechas, horas exactas y periodos de tiempo), hay que destacar que para esta parte se utilizaron expresiones regulares.

Esto es encontrar y extraer estructuras del tipo:

- dd/mm/aa
- dd/mm/aaaa
- aa/mm/dd
- 19 de enero de 2014
- 19/enero/2014
- 19-enero-2014
- Enero 2013
- Enero de 2013
- 12:20 am/pm
- En la tarde, en la mañana, en el mediodía, madrugada, amanecer, etc.

Para realizar la extracción satisfactoria de tiempo se implementaron las siguientes reglas semánticas:

Regla 1
(Mes | (pos=”Card”&tamañopos=2)) (de| / | -) (pos=”Card”& tamañopos=4)

Regla 2
(pos=”Card”& tamañopos=2) (de| / | -) (Mes | pos=”Card”& tamañopos=2) (de| / | -) (pos=”Card”& tamañopos=2 | tamañopos=4)

Regla 3
(pos=”Card”& tamañopos=2) (de| / | -) (Mes | pos=”Card”& tamañopos=2)

Regla 4
(pos=”Card”& tamañopos=2) (:) (pos=”Card”& tamañopos=2) (am| pm | horas)

Regla 5
(pos=”Card”& tamañopos=2) (:) (pos=”Card”& tamañopos=2)(:) (pos=”Card”& tamañopos=2)

Regla 6
(esta | este | la | el) (Mañana | mediodía | tarde | medianoche | madrugada | amanecer)

La implementación de estas reglas se realizaron en Java y su código se puede consultar en el Anexo B.
El resultado obtenido después de ejecutar el módulo de extracción temporal que responde a la pregunta ¿Cuándo ocurrieron los hechos? Se muestra en la Figura 10, dicho resultado se obtiene con la aplicación de la regla 3. La información extraida se utiliza para crear las instancias de la ontología de Tiempo.

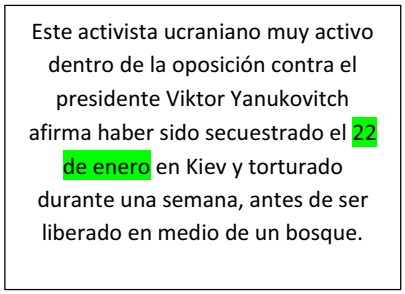

Figura 10. Resultado de extracción temporal

Después de describir todos los módulos que se desarrollarón para elaborar el sistema, en la Figura 11 podemos ver la salida al término de la ejecución de los módulos anteriores con la noticia que se ha manejado en los ejemplos anteriores.

```
----------------------------------------------------------------------NOTICIA: 1----------------------------------------------------------------------
NOTICIA:   Este activista ucraniano muy activo dentro de la oposición contra el presidente Viktor Yanukovitch afirma haber sido secuestrado el 22 d
NOTICIA LIMPIA:   Este activista ucraniano activo de la oposición el presidente Viktor Yanukovitch afirma haber secuestrado el 22 de enero en Kiev
ETIQUETADO:  [?, esto, activista, ucraniano, activo, de, el, oposición, el, presidente, Viktor, Yanukovitch, afirmar, haber, secuestrar, el, 22, de
ETIQUETADO:  [FS, DM, NC, ADJ, ADJ, PREP, ART, NC, ART, NC, NP, NP, VLfin, VHinf, VLadj, ART, CARD, PREP, NMON, PREP, NP, VLadj, NC, PREP, VLadj, P

LA LISTA DE ESPACIOS ENCONTRADOS ES:
en Kiev


LA LISTA DE TIEMPOS ENCONTRADOS ES:
22 de enero
```

Figura 11. Sistema de extracción de información

En la figura 11, se muestra la aplicación de las fases siguientes: limpieza, segmentación, etiquetado y extracción de información.

6.3 Representación

6.3.1 Elaboración del modelo de datos semántico

Para el desarrollo de este módulo fue necesario conocer la taxonomía espacial y temporal, las cuales se implementaron en una ontología en OWL[4] 2.0 para el almacenamiento de las extracciones del módulo anterior.

La Figura 12 muestra la taxonomía de espacio, la cual consta de la siguiente jerarquía: dos clases: espacio natural y espacio urbano. Dentro de espacio natural podemos encontrar las subclases: bosque, montaña, sierra, lago, río, parque y acantilado. Y dentro de espacio urbano están las subclases: plaza, ciudad, escuela, edificio, localidad, colonia, mercado, estado, país y calle. Con una propiedad "Tiene nombre", para enriquecer el modelo de conocimiento, con las extracciones del modulo espacial.

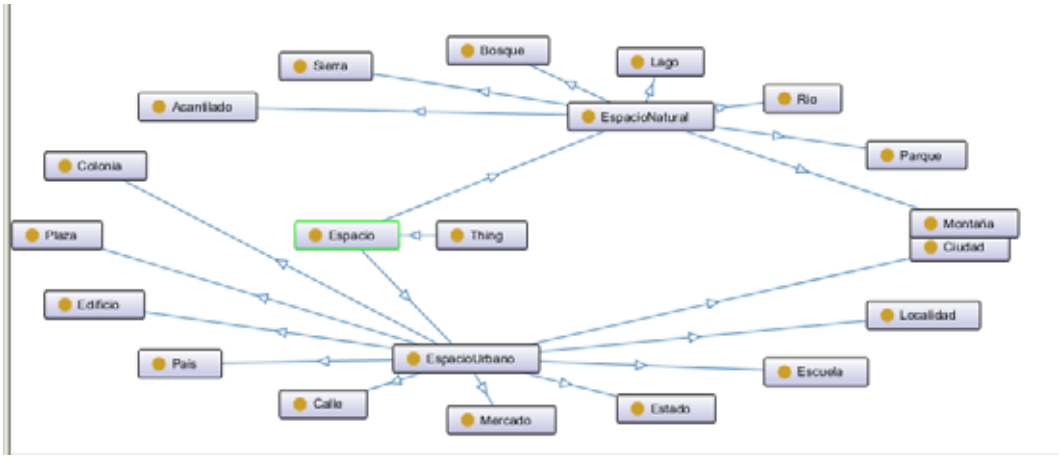

Figura 12. Taxonomía de espacio

La Figura 13 muestra la taxonomía de tiempo, la cual consta de dos clases: hora y fecha. Hora está compuesta por las clases: Hora exacta y Hora periodo. Esta jerarquía de clases va acompañada de la propiedad "Tiene nombre", para en ella enriquecer el modelo de conocimiento y ahí almacenar las extracciones del módulo de temporalidad.

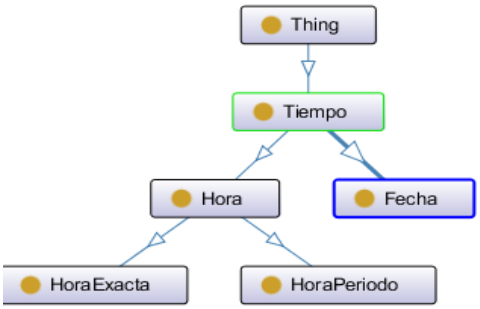

Figura 13. Taxonomía de tiempo

[4] OWL: es el acrónimo del inglés Web Ontology Language, un lenguaje de marcado para publicar y compartir datos usando ontologías

La implementación de este módulo se realizó en XML-OWL 2.0 y su código se puede consultar en el Anexo C.

6.3.2 Poblado Ontológico

En el poblado ontológico utilizamos la información extraida del módulo de extracción espacial y con ella creamos instancias, estas se inserta en la ontología espacio en la clase correspondiente ya sea espacio natural o espacio urbano. En la Figura 14 se puede apreciar la información espacial extraida y su inserción en la clase ciudad, perteneciente a espacio urbano.

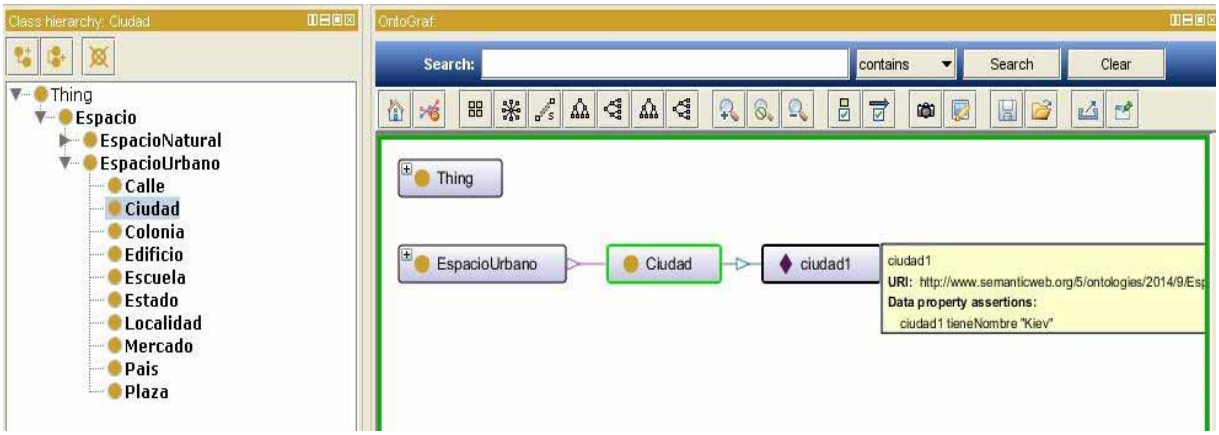

Figura 14. Poblado de ontología espacio.

El código XML generado de la inserción anterior se puede observar en la Figura 15.

```
<ClassAssertion>
    <Class IRI="#Ciudad"/>
    <NamedIndividual IRI="#ciudad1"/>
</ClassAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#tieneNombre"/>
    <NamedIndividual IRI="#ciudad1"/>
    <Literal>Kiev</Literal>
</DataPropertyAssertion>
<DataPropertyDomain>
    <DataProperty IRI="#tieneNombre"/>
    <Class IRI="#Espacio"/>
</DataPropertyDomain>
<DataPropertyRange>
    <DataProperty IRI="#tieneNombre"/>
    <Datatype abbreviatedIRI="xsd:string"/>
</DataPropertyRange>
```

Figura 15. Código XML del poblado de la ontología espacio

En la Figura 16 podemos ver un ejemplo del poblado de la ontología tiempo, especificamente en la clase fecha, para esto se utilizó la información recuperada en el módulo de extracción temporal con la cual se creó la instancia y se inserto en dicha ontología.

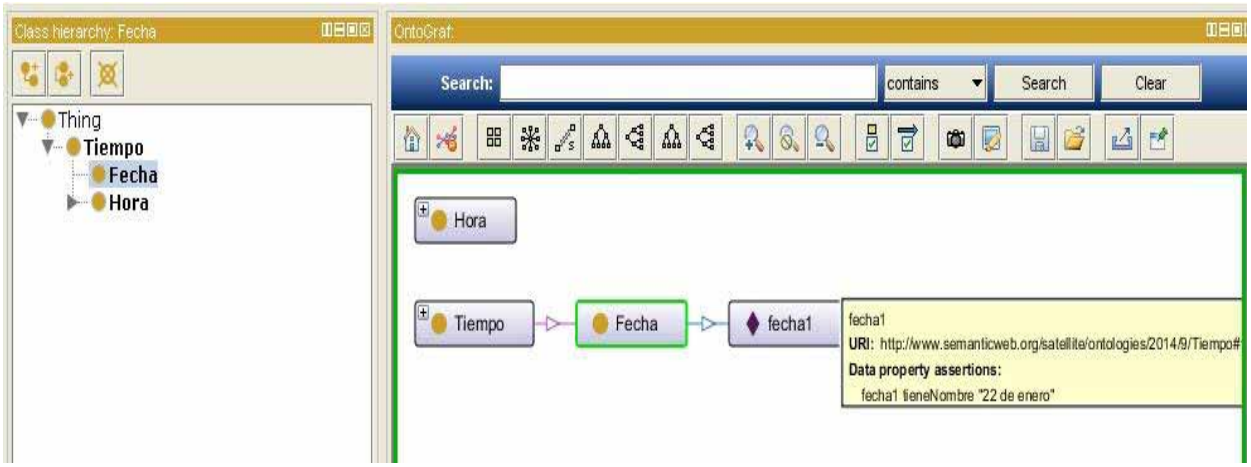

Figura 16. Poblado de ontología tiempo.

Código XML creado con la inserción en la ontología Tiempo, clase fecha (Figura 17).

```
<ClassAssertion>
    <Class IRI="#Fecha"/>
    <NamedIndividual IRI="#fechal"/>
</ClassAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#tieneNombre"/>
    <NamedIndividual IRI="#fechal"/>
    <Literal>22 de enero</Literal>
</DataPropertyAssertion>
<DataPropertyDomain>
    <DataProperty IRI="#tieneNombre"/>
    <Class IRI="#Tiempo"/>
</DataPropertyDomain>
<DataPropertyRange>
    <DataProperty IRI="#tieneNombre"/>
    <Datatype abbreviatedIRI="xsd:string"/>
</DataPropertyRange>
```

Figura 17. Código XML del poblado de la ontología tiempo

7. RESULTADOS

El conjunto de textos para realizar las pruebas fue el establecido en la propuesta esto es, un corpus de 200 noticias nacionales en idioma español, cabe destacar que estas noticias se encuentran en un archivo de texto, donde el formato es: cada línea representa una noticia. En la Figura 18 podemos ver una muestra representativa de dicho corpus de noticias.

Figura 18. Noticias nacionales en idioma español.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y guardar en el modelo de datos semánticos, para el corpus de 200 noticias en idioma español fue de 1 minuto con 25 segundos y se puede corroborar en la figura 19.

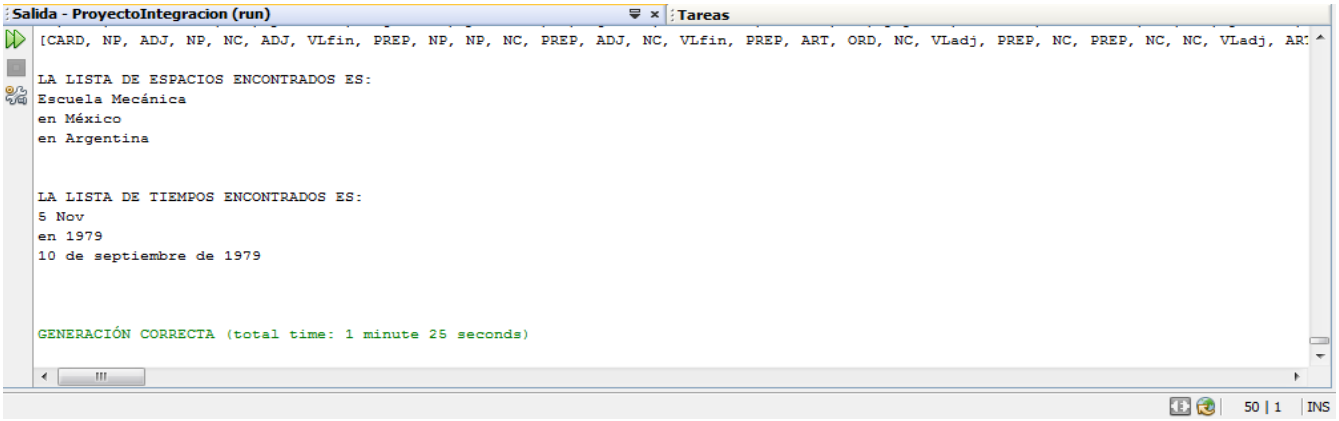

Figura 19. Tiempo de procesamiento.

7.1 Resultados de la extracción espacial.

En la Tabla 2 podemos ver todos los espacios extraídos de los 200 textos periodísticos, con un total encontrado de **376** espacios y un promedio por noticia de **1.88**, estos espacios responden a la pregunta ¿Dónde sucedió el evento?, algunos de estos eventos los podemos ver instanciados en la ontología espacio en la Figura 20.

$$\boldsymbol{Número\ total\ de\ espacios\ encontrados\ \ \omega = 376}$$

$$Número\ de\ espacios\ en\ \ promedio\ por\ noticia = \frac{espacios\ extraídos}{número\ total\ de\ noticias}$$

$$Número\ de\ espacios\ en\ promedio\ por\ noticia = \frac{376}{200}$$

$$Promedio\ de\ lugares\ encontrados\ por\ noticia = 1.88\ \approx 2.0\ lugares$$

7.2 Resultados de la extracción temporal.

En la Tabla 3 podemos ver los resultados extraídos con información temporal (¿Cuándo ocurrió?) de los 200 textos periodísticos, con un número total de **133** y un promedio de **0.66** por noticia. En la figura 21 se pueden apreciar algunos de estos resultados instanciados en la ontología de tiempo en la clase de hora exacta.

$$\boldsymbol{Número\ total\ de\ informacion\ temporal\ encontrada\ \ \gamma = 133}$$

$$Información\ temporal\ obtenida\ en\ \ promedio\ por\ noticia = \frac{temporales\ extraídos}{número\ total\ de\ noticias}$$

$$Información\ temporal\ obtenida\ en\ \ promedio\ por\ noticia = \frac{133}{200}$$

$$Información\ temporal\ obtenida\ \ en\ \ promedio\ por\ noticia = 0.66 \approx 1.0\ lugar$$

RESULTADOS DE EXTRACCIÓN ESPACIAL							
en Kiev	vía Bogotá-Choachí	en Ishikawa	en Tokio	en Ebon	en Manila	en Julio	en Camarena
isla Marshall	isla Phillip	isla Muller	estado Unidos	cadena CNN	en Bosnia	en Ginebra	en Vieja
ciudad Vieja	ciudad Homs	casa Blanca	casa Jay	casa Carney	casa Blanca	en Ishikawa	en Washington
en Santo	en Francisco	estado Unidos	en Johannesburgo	en Colombia	en calle carretera	ciudad Apatzingán	en Dinamarca
en Irán	en Raposo	en casa padre	en Salvador	en Salvador	en Parlamento	en Parlamento	en Kiev
en Ucrania	en isla	isla Marshall	en México	instituto Geofísico	instituto Fernanda	instituto Naranjo	instituto Memoria
instituto Memoria	estado Unidos	en Plaza	Plaza Independencia	Plaza Independencia	en país triángulo	en Majuro	municipio Guanay
en Washington	ciudad Nueva	ciudad York	casa Francia	en Habana	país Finlandia	país Singapur	país Israel
en Janeiro	en Brasil	zona Japón	zona Japón	en Inglaterra	isla Sajalín	isla Sajalín	en Chile
en Chile	en Chiba	en isla	isla Marshall	isla Marshall	país América	país Latina	en América
en Johannesburgo	en Oriental	en provincia cabo	cumbre Cali	en Cali	localidad Fusagasugá	localidad Silvania	vía Bogotá-Choachí
localidad Chullpa	localidad K'asa	en Moscú	en Sudáfrica	en Cartagena	municipio Tomatlán	en Mora	municipio Tomatlán
en Filipinas	en Julio	en Camarena	pueblo Garita	pueblo Palmera	pueblo Garita	pueblo Palmera	en AIEA
en Gachin	en AIEA	localidad Ch'ullpa	municipio Morochata	municipio Morochata	zona Japón	montaña Ourense	en municipio montaña
estado Caribeños	en Nelson	en Tíbet	instituto Geofísico	colegio Santo	colegio Martín	en Efe	provincia Tungurahua
provincia Tungurahua	provincia Chimborazo	pueblo México	pueblo Chile	pueblo México	pueblo Colombia	isla Marshall	en Océano
en Pacífico-	Océano Pacífico-	pueblo Garita	pueblo Palmera	pueblo Garita	pueblo Palmera	en Vieja	ciudad Vieja
ciudad Homs	en Alvarenga	en Bruselas	en Dinamarca	en Euromaidán	en Mezhigorie	ciudad Vieja	ciudad Homs
ciudad Homs	en Suiza	en Efe	en Johannesburgo	en provincia Cabo	provincia Cabo	Río Rocha	zona Muyurina
en Nepal	en Qunu	en Qunu	en Bandar	en Abás	en Irán	en Bandar	en Abás
en Rica	en Vaticano	estado Unidos	cadena CNN	ciudad Cartagena	en Qunu	barrio Houghton	en Munich
en Kerry	cumbre Mujica	municipio Morochata	en Chullpa	municipio Morochata	en Cochabamba	Colegio Santo	Colegio Martín
Colegio Santo	Colegio Martín	localidad Fusagasugá	localidad Silvania	municipio Apatzingán	municipio Apatzingán	país Finlandia	país Singapur
país Israel	municipio Tonalá	en Chiapas	en Azul	en Tonalá	en AIEA	en Gachin	en Teherán
casa Rey	Estado Colombia	en Cartagena	país Colombia	construcción Civil	municipio Morochata	en Morochata	en Tiempos
en Chullpa	en K´asa	en Chico	municipio Morochata	en Hanoi	en Vietnam	en Hanoi	en Vietnam
en AIEA	instituto Geofísico	isla Sajalín	en TLC	en México	en Bandar	en Abás	en Humala
cumbre Colombia	cumbre Piñera	en Colombia	en Piñera	en Humala	cumbre Colombia	en Colombia	en Colombia
isla Marshall	calle Kiev	en Kiev	en Kiev	en Rurales	en Rurales	cumbre Mujica	cumbre Mujica
casa Rey	en Alvarenga	en Alvarenga	en Apatzingán	en Morelia	zona Apatzingán	estado Unidos	municipio Apatzingán
en Unidos	en estado Unidos	estado Unidos	en Vieja	ciudad Vieja	ciudad Homs	en Janeiro	en Chile
estado Caribeños	en Cuba	en Tokyo	cumbre Mujica	en Brasil	en Siria	en Brasil	en Sao
en Paulo	municipio Apatzingán	isla Puerto	ciudad Santo	ciudad Salvador	ciudad Santa	ciudad Ana	ciudad Santo
ciudad Salvador	ciudad Santa	ciudad Uppsala	ciudad Suecia	en Pekín	país Vasco	País Vasco	isla Terranova
provincia León	Vía plata	en Boston	capital Michoacán	en Brooklyn	en Warwick	en Wallkill	en Brooklyn
en Nueva	en York	en Amecameca	en Cenapred	pueblo Santa	pueblo Bárbara	en Cuautitlán-Teoloyucan	en municipio Navolato
en municipio Santo	municipio Santo	en municipio riesgo	en zona riesgo	en Culiacán	en Navolato	en Elota	municipio Escuinapa
Villa Fueron	en vivienda colonia	colonia Santa	colonia María	colonia Aztahuacán	en Iztapalapa	ciudad México	Escuela Normal
Escuela Rural	Escuela Raúl	calle Cedro	Villa Fueron	en vivienda colonia	colonia Santa	colonia María	colonia Aztahuacán
en Iztapalapa	en Aires	en Kirchner	en Bisbane	en Iguala	en Teotihuacan	país Aquella	en barrio élite
en barrio clase	en Matemáticas	en Sistemas	en Cretácico	en Superior	en Madagascar	en Gondwana	en altura mandíbula
en China	país Según	en China	Escuela Pública	en Subcentro	en Selva	en Xtomoc	en Quintana
en Roo	en Guayana	en Francesa	municipio Santo	municipio Ignacio	en Coyoacán	en Ex	en Francisco
en Sosa	Barrio Santa	Barrio Catarina	Barrio Coyoacán	en México	colegio Humanidades	en Morelia	Estado Michoacán
escuela Normales	municipio Guerrero	en Argentina	provincia Aires	localidad Santo	localidad Fernando	ciudad Aires	cadena CNN
cadena HRW	en Kobane	en Siria	en La	en Organización	país América	país Latina	instituto Carlos
en Chihuahua	municipio Buenaventura	en Cuauhtémoc	en Armandina	en Márquez	Escuela Mecánica	en México	en Argentina

Tabla 2. Resultados de la extracción espacial

RESULTADOS DE EXTRACCIÓN TEMPORAL							
5 Nov	en 1979	10 de septiembre de 1979	5 Nov	18:42:53	2014-11-04	22:57:32	2014-11-05
22:20:36	22:49:46	28 de octubre	14:25:30	26 de septiembre	08:28:20	10:52:27	04:00
2014-11-05	01:03:10	30 de noviembre	11:38:08	en 2012	en 2012	en 2012	12:16:26
2014-11-05	18:22:03	5 de noviembre de 2014	2:30	05 nov	05 nov 2014	nov 2014	10:03
el mañana	16 de septiembre	21 de octubre	6 de octubre	20:03	01:30	10 de noviembre de 2084	24 de febrero de 1836
en 1979	en 1977	en 1492	en 1453	18 de marzo de 1938	26 de julio de 1325	en 1982	en 2013
en 2014	3 de febrero de 1989	12 de febrero	29 de enero	6 de abril	27 de enero	diciembre de 2010	desde 2006
para 2014	19 de marzo	19 de marzo	abril de 2010	14:00	6 de abril	desde 2012	desde 2012
en 2005	8 de febrero	11 de febrero	en 1948	en 1996	este mañana	2:30	2:30 pm
19 de marzo	9 feb	21 de noviembre	desde 2012	doce de 31	2014-2018	8 de mayo	8 de mayo
6 de abril	9 de marzo	9 feb	24 de diciembre de 2012	21 de febrero	en 1948	15:30	12 de octubre de 2004
en 2008	en 2005	11 de febrero	11 de febrero	9 de febrero	2:30	2:30 pm	en 2013
12 de octubre de 2004	en 2008	5 de diciembre	12 de octubre de 2004	en 2008	30 de enero	26 de noviembre	en 1966
7 feb	el mañana	11 de febrero de 2013	11 de febrero	11 de febrero del 2013	del 2013	19:30	noviembre de 2012
7 feb	desde 1996	10-02-2014	25 de mayo	24 de diciembre de 2012	21 de diciembre de 2012	del 2004	en 2005
10 feb	marzo de 2012	31 de enero	29 de enero	desde 1996	22 de enero	Febrero	Febrero 10
Febrero 10 de 2014	10 feb	desde 2012					

Tabla 3. Resultados de la extracción temporal

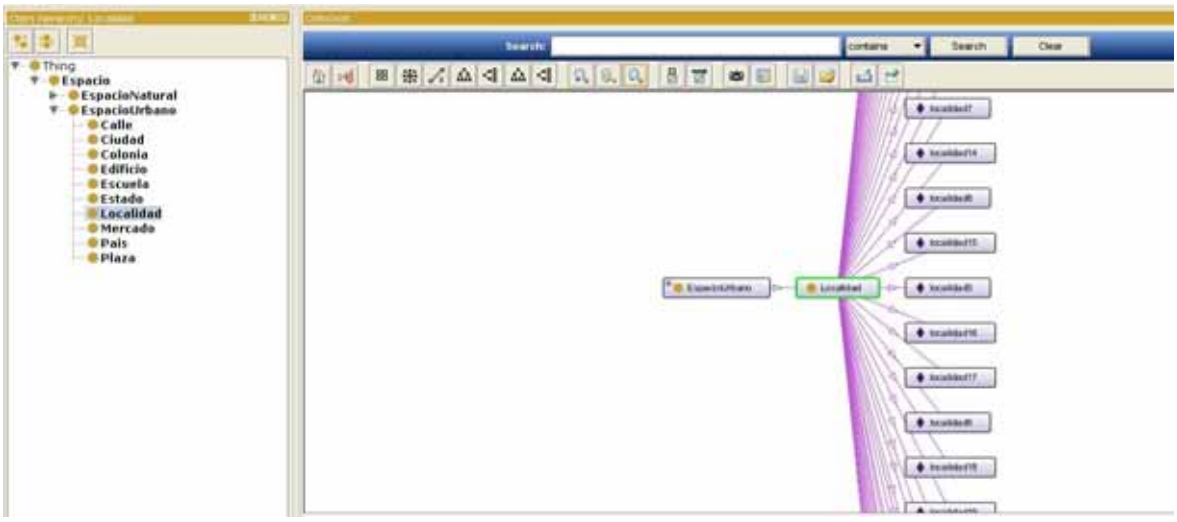

Figura 20. Instancias de la ontología Espacio

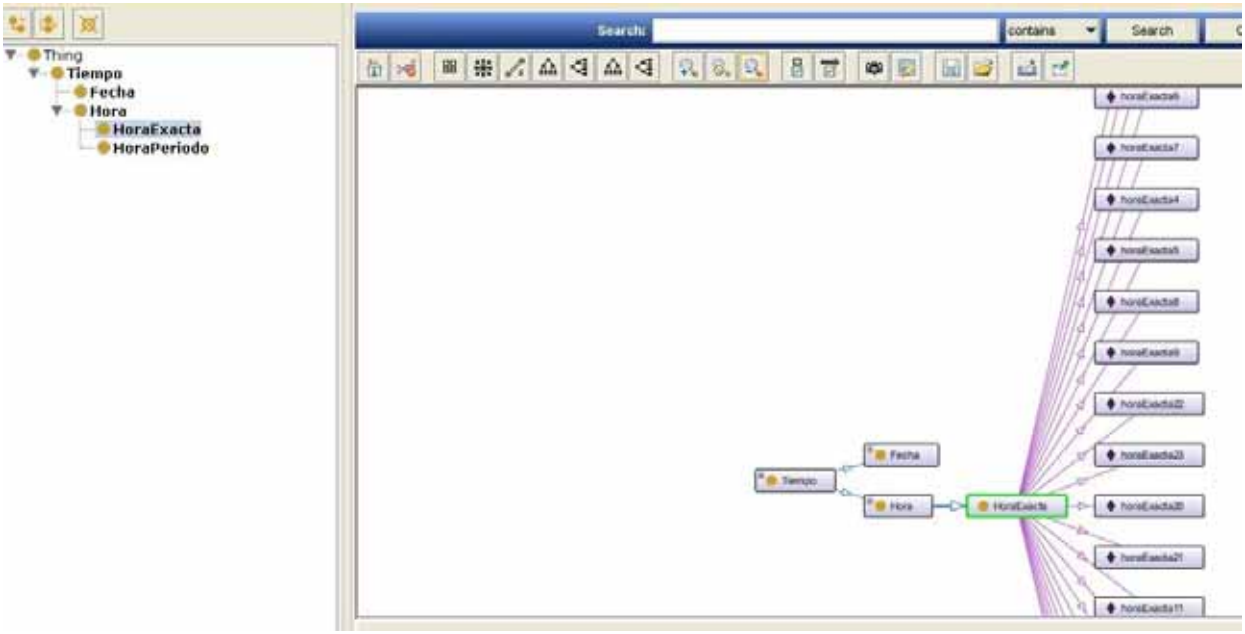

Figura 21. Representación en la ontología tiempo

8. ANÁLISIS Y DISCUSIÓN DE RESULTADOS

8.1 Análisis de la extracción de espacios

Para la evaluación de la tarea de extracción de información espacial a partir de los textos periodísticos, tenemos que analizar la precisión con la que se obtuvieron los resultados, esto es, revisar el número de espacios correctos y el número de espacios erróneos.

En la Tabla 4 podemos encontrar los elementos erróneos extraídos estos nos dan un número total de 65, luego el número total de espacios correctos es de 311. Con la ayuda de estos datos a continuación calcularemos la precisión del sistema.

$$\boldsymbol{Elementos\ extraídos = 376}$$

$$\boldsymbol{Elementos\ Relevantes\ Extraidos = 311}$$

$$Precisión\ = \frac{elementos\ relevantes}{elementos\ extraidos}\ x\ 100$$

$$Precisión\ = \frac{311}{376}\ x\ 100$$

$$Precisión\ = \mathbf{82.71}\ \%$$

Como podemos apreciar el desempeño de la técnica de extracción espacial en textos de noticias es alto puesto que se obtiene un error del 17.29% (66 entidades de 376); el tiempo de ejecución también está dentro de los paramentos aceptables.

RESULTADOS EXTRAÍDOS NO RELEVANTES						
en Julio	en Camarena	isla Phillip	isla Muller	cadena CNN	casa Jay	casa Carney
en calle carretera	en Raposo	en Parlamento	en Parlamento	instituto Fernanda	instituto Naranjo	en país triángulo
en Oriental	en Mora	en Julio	en Camarena	en municipio montaña	en Nelson	en Efe
en Alvarenga	en Euromaidán	en Efe	cadena CNN	en Kerry persona	cumbre Mujica	en Azul costa azul
en Tiempos	en TLC	cumbre Piñera	en Piñera	en Rurales	en Rurales	cumbre Mujica
cumbre Mujica	en Alvarenga	en Alvarenga	en municipio riesgo	en zona riesgo	Villa Fueron	en vivienda colonia
Villa Fueron	en Kirchner	país Aquella	país Aquella	en barrio élite	en barrio clase	en Matemáticas
en Sistemas	en Cretácico	en Superior	en altura mandíbula	país Según	en Subcentro	en Ex
en Francisco	en Sosa	lago Planetario	cadena CNN	cadena HRW	en La	en Organización
en Armandina	en Márquez					

Tabla 4. Lugares no relevantes (errores)

8.2 Análisis de la extracción temporal

De manera similar ahora vamos a analizar los resultados de la extracción temporal para ello vamos a describir los valores obtenidos y son los siguientes:

$\boldsymbol{Elementos\ Erroneos\ Extraidos = 1}$ (Tabla 5)

$\boldsymbol{Elementos\ Relevantes\ Extraidos = 132}$

$$Precisión\ = \frac{elementos\ relevantes}{elementos\ extraidos}\ x\ 100$$

$$Precisión\ = \frac{132}{133}\ x\ 100$$

$$Precisión\ = \mathbf{99.24}\ \%$$

Como se puede apreciar el porcentaje indica una alta eficiencia de la técnica de extracción temporal ya que solo se obtiene un error de 0.76%.

RESULTADOS TEMPORALES EXTRAÍDOS NO RELEVANTES
doce de 31

Tabla 5. Tiempos no relevantes

9. CONCLUSIONES

Como conclusión de este proyecto de integración podemos decir que el objetivo primordial que era extraer información espacial y temporal a partir de textos periodísticos descritos en español aplicando reglas lingüísticas con la finalidad de enriquecer un modelo de datos semántico, se ha cumplido satisfactoriamente, sin embargo no toda la extracción ha sido relevante porque en la técnica de extracción espacial se encontraron deficiencias, que generaron errores del 17.29 %, estos se pueden explicar debido a que el etiquetador utilizado presentaba algunas inconsistencias; por ejemplo: algunas veces no se podía diferenciar de un nombre propio a un nombre común, tal vez esto ocurra ya que dicho etiquetador esta optimizado para el idioma inglés, algunos otros errores se dieron por la polisemia o ambigüedad en las palabras. Otros tantos por las reglas lingüísticas, en especial la regla 1 ya que un nombre propio puede ser el de una persona o el de una entidad y ahí se encuentra un conflicto con lo que esperamos obtener. Sin embargo, se logró una efectividad del 82.71 % en la extracción espacial, en cuanto a la extracción temporal se logró una efectividad del 99% en gran parte al manejo de una herramienta muy poderosa como lo son las expresiones regulares, estos resultados son muy buenos ya que dada la relevancia que tiene hoy en día el computo inteligente y el aprendizaje máquina, podemos optimizar el tiempo que emplea el ser humano en la búsqueda de información. En base a los resultados obtenidos podemos decir que este proyecto es una herramienta confiable.

Finalmente, se considera que la relevancia de este proyecto se debe a la importancia que tiene la extracción de información, porque es una herramienta de gran utilidad que sirve para búsqueda de información, nos ayuda a responder preguntas como ¿cuándo ocurrió el hecho y donde ocurrió?; como se llevó a cabo este trabajo.

Los resultados de este proyecto de integración pueden ser mejorados en un futuro, con la utilización de otro etiquetador, reestructurando algunas de las reglas lingüísticas implementadas u optimizando el código de programación.

10. ANEXOS

10.1 Anexo A: Código fuente de la limpieza de textos periodísticos

```
package Procesamiento;

/**

 *

 * @author Josué Padilla Cuevas  */

public class Limpieza {

String limpio;
public String Limpia(String PalabrasVacias[], String noticia){
   String guarda="";
   String nuevo=noticia;
   for (int i = 0; i < PalabrasVacias.length; i++) {
         guarda =" "+PalabrasVacias[i]+" ";


      // quita todas las palabras vacias del idioma español en las Noticias nacionales

            nuevo= nuevo.replaceAll(guarda, " ");
            nuevo=nuevo.replace(".", "");
            nuevo=nuevo.replace(",", "");
            nuevo=nuevo.replace(";", "");


      }

               limpio=nuevo;



   return limpio;

} }
```

10.2 Código fuente del módulo de segmentación y etiquetado.

```
package Procesamiento;

import java.io.IOException;
import org.annolab.tt4j.*;
import java.util.ArrayList;
import static java.util.Arrays.asList;
/**

 *

 * @author Josué Padilla Cuevas

 */

public class Etiquetado {

   ArrayList<ArrayList<String>> pEtiqueta = new ArrayList<ArrayList<String>>();
   int tamaño;


   public int getTamaño() {
      return tamaño;
   }

     public ArrayList<ArrayList<String>> EiquetaPos(String NoticiaLimpia) throws IOException,
TreeTaggerException {



     // declaración de los arreglos donde guardaremos las etiquetas
      String[] cadenaEntrada = NoticiaLimpia.split(" ");
      ArrayList<String> post = new ArrayList<String>();
      ArrayList<String> lema = new ArrayList<String>();
      ArrayList<String> tokn = new ArrayList<String>();



      // Point TT4J to the TreeTagger installation directory. The executable is expected

      // in the "bin" subdirectory - in this example at "/opt/treetagger/bin/tree-tagger"

      System.setProperty("treetagger.home", "C:/TreeTagger");
      TreeTaggerWrapper<String> tt = new TreeTaggerWrapper<String>();


      try {
```

```
   // lugar donde esta almacenada la aplicación TreeTagger
   tt.setModel("C:/TreeTagger/modelos/spanish.par:iso8859-1");
   tt.setHandler(new TokenHandler<String>() {
      public void token(String token, String pos, String lemma) {


         // guardar en los arreglos las etiquetas
          lema.add(lemma);
          post.add(pos);
          tokn.add(token);


      }

   });


   tt.process(asList(cadenaEntrada));
} finally {
   tt.destroy();
}
tamaño = lema.size();


//recupera las fechas que se pierden durante el etiquetado
for (int i = 0; i < tamaño; i++) {
   if (lema.get(i).equals("@card@")) {
      lema.set(i, tokn.get(i));
   }

}

// guarda en el arreglo de arreglos
pEtiqueta.add(lema);
pEtiqueta.add(post);
 System.out.println("ETIQUETADO:  "+lema);
System.out.println("ETIQUETADO:  "+post);
return pEtiqueta;    }}
```

10.3 Anexo B: Código fuente del módulo de extracción temporal.

```
package Procesamiento;

import Ontologia.InsertInstance;
import java.util.ArrayList;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyStorageException;


/**

 *

 * @author Josué Padilla Cuevas

 */

public class ExtraccionTiempo {

   int a = 1, b = 1, c = 1;

   public String ReglasTiempo(ArrayList<ArrayList<String>> t, int tamaño) throws
OWLOntologyStorageException, OWLOntologyCreationException {

      String extraer = "";
      String enviar = "";
      InsertInstance in = new InsertInstance();
      String ont = "Tiempo.owl";
      String iri = "http://www.semanticweb.org/satellite/ontologies/2014/9/Tiempo#";
      String prop = "tieneNombre";


     // Expresiones regulares para encontrar fechas y horas

      String fecha = "[\\d]{1,4}[/|-][\\d]{1,4}[/|-]{0,1}[\\d]{0,4}";
      String fecha1 = "[\\d]{0,2}[/|-]{0,1}[a-zA-Z]{3,10}[/|-][\\d]{2,4}";
      String hora = "[\\d]{1,2}[:][\\d]{1,2}[:]{0,1}[\\d]{0,2}";


      //declaracion de los meses del año
      String[] Mes = {"enero", "febrero", "marzo", "abril", "mayo", "junio", "julio",
         "agosto", "septiembre", "octubre", "noviembre", "diciembre",
         "ene", "feb", "mar", "abr", "may", "jun", "jul",
         "agosto", "sep", "oct", "nov", "dic"};
```

```
        // declaracion de los días de la semana
        String[] Dia = {"lunes", "martes", "miercoles", "jueves", "viernes"};


        for (int i = 0; i < tamaño; i++) {

           //recupera las posiciones en el arreglo de arreglos
           String lema = "", pos = "", lema1 = "", pos1 = "", lema2 = "", pos2 = "", lema3 = "",
                 pos3 = "", lema4 = "", pos4 = "";
           lema = t.get(0).get(i).toString();
           pos = t.get(1).get(i).toString();


           // Variables que evitan el desbordamiento del array
           if (i < tamaño - 1) {
              pos1 = t.get(1).get(i + 1).toString();
              lema1 = t.get(0).get(i + 1).toString();
           }
           if (i < tamaño - 2) {
              pos2 = t.get(1).get(i + 2).toString();
              lema2 = t.get(0).get(i + 2).toString();
           }
           if (i < tamaño - 3) {
              pos3 = t.get(1).get(i + 3).toString();
              lema3 = t.get(0).get(i + 3).toString();
           }
           if (i < tamaño - 4) {

              pos4 = t.get(1).get(i + 4).toString();
              lema4 = t.get(0).get(i + 4).toString();
           }


//Reglas lingüisticas para encontrar fechas
        // Regla para encontrar fechas del tipo Enero de 2014
        for (int j = 0; j < Mes.length; j++) {
           if (lema.equalsIgnoreCase(Mes[j])) {
              if ((i > 0 && !t.get(0).get(i - 1).toString().equalsIgnoreCase("de"))
                    && ((lema1.equalsIgnoreCase("de")) || (lema1.equalsIgnoreCase("/")) ||
(lema1.equalsIgnoreCase("-")))
                    && (((pos2.equalsIgnoreCase("CARD") || pos2.equalsIgnoreCase("CODE"))

                    && (lema2.length() == 4 || lema2.length() == 2))) {
```

```
                extraer += lema + " " + lema1 + " " + " " + lema2 + "\n";
                enviar = lema + " " + lema1 + " " + " " + lema2;
                in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
                in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
                a++;

            }


            if (((pos1.equalsIgnoreCase("CARD") || (pos1.equalsIgnoreCase("CODE")))
                && (lema1.length() == 4 || lema1.length() == 2)) &&
!lema2.equalsIgnoreCase("de")) {
                enviar = lema + " " + lema1;
                extraer += lema + " " + lema1 + "\n";
                in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
                in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
                a++;
            }
          }
        }

        // Regla para encontar fechas de la forma Febrero 10 de 2014
        for (int j = 0; j < Mes.length; j++) {
          if ((lema.equalsIgnoreCase(Mes[j]))
                && (((pos1.equalsIgnoreCase("CARD") || (pos1.equalsIgnoreCase("CODE"))) &&
lema1.length() == 2))
                && (lema2.equalsIgnoreCase("de"))
                && ((pos3.equalsIgnoreCase("CARD") || pos1.equalsIgnoreCase("CODE")) &&
lema3.length() == 4)) {
            extraer += lema + " " + lema1 + " " + lema2 + " " + lema3 + "\n";
            enviar = lema + " " + lema1 + " " + lema2 + " " + lema3;
            in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
            in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
            a++;
          }

        }



        //regla para encontrar fechas del tipo 12/20/1988 o 12/20/88
        if (lema.matches(fecha) || lema.matches(fecha1)) {
          extraer += lema + "\n";
```

```
            enviar = lema;
            in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
            in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
            a++;
          }


          //Regla 3 para encontrar fechas del tipo 22 de diciembre o 22 de diciembre de 2014
        if (((pos.equalsIgnoreCase("CARD") || pos.equalsIgnoreCase("CODE")) && lema.length() <= 4)
              && ((lema1.equalsIgnoreCase("de")) || (lema1.equalsIgnoreCase("/"))
              || (lema1.equalsIgnoreCase("-")))) {

            if (((pos2.equalsIgnoreCase("CARD") || pos2.equalsIgnoreCase("CODE")) &&
lema2.length() == 2)) {
              if (!lema3.equalsIgnoreCase("de")) {
                extraer += lema + " " + lema1 + " " + lema2 + "\n";
                enviar = lema + " " + lema1 + " " + lema2;
                in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
                in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
                a++;
              }
              if ((lema3.equalsIgnoreCase("de") || lema3.equalsIgnoreCase("del")

                    || lema3.equalsIgnoreCase("/") || lema3.equalsIgnoreCase("-"))
                    && (((pos4.equalsIgnoreCase("CARD") || pos4.equalsIgnoreCase("CODE"))
                    && lema4.length() <= 4))) {
                extraer += lema + " " + lema1 + " " + lema2 + " " + lema3 + " " + lema4 + "\n";
                enviar = lema + " " + lema1 + " " + lema2 + " " + lema3 + " " + lema4;
                in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
                in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
                a++;
              }

            }

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

              if (lema2.equalsIgnoreCase(Mes[j])) {
                if (!lema3.equalsIgnoreCase("de")) {
                  extraer += lema + " " + lema1 + " " + lema2 + "\n";
                  enviar = lema + " " + lema1 + " " + lema2;
                  in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
                  in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
                  a++;                    }
```

```
        if ((lema3.equalsIgnoreCase("de") || lema3.equalsIgnoreCase("del")

                || lema3.equalsIgnoreCase("/") || lema3.equalsIgnoreCase("-"))
                && (((pos4.equalsIgnoreCase("CARD") || pos4.equalsIgnoreCase("CODE"))
                && lema4.length() <= 4))) {
            extraer += lema + " " + lema1 + " " + lema2 + " " + lema3 + " " + lema4 + "\n";
            enviar = lema + " " + lema1 + " " + lema2 + " " + lema3 + " " + lema4;
            in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
            in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
            a++;
        }

      }

   }

}

//Regla para extraer fechas de la manera 22 enero de 2014 o 22 enero 2014 o 22 enero

if (((pos.equalsIgnoreCase("CARD") || pos.equalsIgnoreCase("CODE"))

      && (lema.length() == 2) || lema.length() == 1)) {
   for (int j = 0; j < Mes.length; j++) {
      if (lema1.equalsIgnoreCase(Mes[j])) {
         if ((!lema2.equalsIgnoreCase("de")) || !lema2.equalsIgnoreCase("del")) {
            extraer += lema + " " + lema1 + "\n";
            enviar = lema + " " + lema1;
            in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
            in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
            a++;
         }
         if ((lema2.equalsIgnoreCase("de")) || lema2.equalsIgnoreCase("del")

                && ((pos3.equalsIgnoreCase("CARD") || pos3.equalsIgnoreCase("CODE"))
                && (lema3.length() == 2
                || lema3.length() == 4))) {
            enviar = lema + " " + lema1 + " " + lema2 + " " + lema3;
            extraer += lema + " " + lema1 + " " + lema2 + " " + lema3 + "\n";
            in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
            in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
            a++;

         }
```

```
        if ((pos2.equalsIgnoreCase("CARD") || pos2.equalsIgnoreCase("CODE"))
            && (lema2.length() == 2 || lema2.length() == 4)) {
          extraer += lema + " " + lema1 + " " + lema2 + "\n";
          enviar = lema + " " + lema1 + " " + lema2;
          in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
          in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
          a++;
        }

      }

  }

}



// Regla para encontrar fechas del tipo En, para, desde, de, del 2012

if ((lema.equalsIgnoreCase("en")
      || lema.equalsIgnoreCase("para") || lema.equalsIgnoreCase("desde")
      || lema.equalsIgnoreCase("del")) && ((pos1.equalsIgnoreCase("CARD")
      || (pos1.equalsIgnoreCase("CODE")))
      && (lema1.length() == 4))) {
  extraer += lema + " " + lema1 + "\n";
  enviar = lema + " " + lema1;
  in.createClassIndividual(ont, iri, "Fecha", "fecha" + a);
  in.createDataPropertyAssertion(ont, iri, "fecha" + a, prop, enviar);
  a++;
}



// Reglas para encontrar las horas enlas noticias

//Regla1 para encontrar las horas exactas en las noticias
if (lema.matches(hora)) {
  if (!lema1.equalsIgnoreCase("horas") || !lema1.equalsIgnoreCase("am")
      || !lema1.equalsIgnoreCase("pm") || !lema1.equalsIgnoreCase("hrs")) {
    extraer += lema + "\n";
    enviar = lema;
    in.createClassIndividual(ont, iri, "HoraExacta", "horaExacta" + b);
    in.createDataPropertyAssertion(ont, iri, "horaExacta" + b, prop, enviar);
    b++;
  }       }
```

```
        if (lema.matches(hora)) {
           if (lema1.equalsIgnoreCase("horas") || lema1.equalsIgnoreCase("am")
                 || lema1.equalsIgnoreCase("pm") || lema1.equalsIgnoreCase("hrs")) {
              extraer += lema + " " + lema1 + "\n";
              enviar = lema + " " + lema1;
              in.createClassIndividual(ont, iri, "HoraExacta", "horaExacta" + b);
              in.createDataPropertyAssertion(ont, iri, "horaExacta" + b, prop, enviar);
              b++;
           }

        }

        //Regla 2 para encontrar las horas periodo ejemplo esta tarde
        if ((lema.equalsIgnoreCase("esta") || lema.equalsIgnoreCase("este")
              || lema.equalsIgnoreCase("la") || lema.equalsIgnoreCase("el"))
              && (lema1.equalsIgnoreCase("mañana") || lema.equalsIgnoreCase("mediodía")
              || lema.equalsIgnoreCase("tarde") || lema.equalsIgnoreCase("medianoche")
              || lema.equalsIgnoreCase("madrugada") || lema.equalsIgnoreCase("amanecer"))) {
           extraer += lema + " " + lema1 + "\n";
           enviar = lema + " " + lema1;
           in.createClassIndividual(ont, iri, "HoraPeriodo", "horaPeriodo" + c);

           in.createDataPropertyAssertion(ont, iri, "horaPeriodo" + c, prop, enviar);
           c++;
        }

//Regla de horas

        if (((pos.equalsIgnoreCase("CARD") || pos.equalsIgnoreCase("CODE"))
              && lema.length() == 2) && (lema1.equalsIgnoreCase("horas")
              || lema1.equalsIgnoreCase("minutos"))) {
           enviar = lema + " " + lema1;
           extraer += lema + " " + lema1 + "\n";
            in.createClassIndividual(ont, iri, "HoraPeriodo", "horaPeriodo" + c);
            in.createDataPropertyAssertion(ont, iri, "horaPeriodo" + c, prop, enviar);
           c++;
        }

     }

     return extraer;

  }}
```

10.4 Código fuente del módulo de extracción espacial.

```
package Procesamiento;

import Ontologia.InsertInstance;
import java.util.ArrayList;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyStorageException;


/**

 *

 * @author josué Padilla cuevas

 */

public class ExtraccionEspacio {
 int b=1,c=1,d=1,e=1,f=1,g=1,h=1,k=1,l=1, m=1, n=1,o=1,p=1,q=1,r=1,s=1,u=1;
   public ExtraccionEspacio() {

   }

   public String AplicarReglas(ArrayList<ArrayList<String>> t, int tamaño) throws
OWLOntologyStorageException, OWLOntologyCreationException {
      String extraer = "";
      String enviar = "";
      InsertInstance in = new InsertInstance();
      String ont = "Espacio.owl";
      String iri = "http://www.semanticweb.org/5/ontologies/2014/9/Espacio#";
      String prop="tieneNombre";

      // Marcador Lingüistico Urbano
      String[] Plaza = {"Plaza", "plaza"};
      String[] Ciudad = {"ciudad", "Metrópolis", "Centro urbano", "capital"};
      String[] Escuela = {"Escuela", "colegio", "academia", "instituto", "liceo", "conservatorio",
"facultad"};
      String[] Edificio = {"Edificio", "inmueble", "casa", "vivienda", "obra", "bloque", "finca",
"construcción", "edificación", "fábrica", "manzana"};
      String[] Localidad = {"localidad", "Municipio", "población", "aldea", "pueblo", "villa"};

      String[] Colonia = {"Colonia", "Barrio", "condominio"};
```

```
        String[] Mercado = {"Mercado", "feria", "mercadillo", "bazar", "supermercado"};
        String[] Estado = {"estado", "jurisdicción", "provincia", "zona"};
        String[] Pais = {"País", "Nación", "territorio"};
        String[] Calle = {"Calle", "vía", "paseo", "avenida", "boulevard", "ronda", "carrera", "callejón",
"travesía", "pasadizo", "pasaje", "calzada"};
       // fin de Marcador Urbano


       // Marcador Lingüistico Natural
        String[] Bosque = {"Bosque","selva", "espesura", "frondosidad", "boscaje", "arbolado",
"huerto"};
        String[] Montaña = {"Montaña", "cordillera", "cadena", "monte", "cumbre", "cima",
"prominencia", "promontorio", "altura", "elevación", "cerro", "colina", "montículo", "loma"};
        String[] Sierra = {"Sierra","pico", "serranía"};
        String[] Lago = {"Lago", "albufera", "estanque", "laguna", "marisma", "alberca", "pantano",
"balsa", "charca", "estero"};
        String[] Rio = {"río", "torrente", "arroyo", "torrentera", "riachuelo", "riacho", "afluente",
"regato", "abundancia", "caudal", "profusión", "raudal", "cantidad", "plétora",
"barranco","Océano"};
        String[] Parque = {"Parque", "Jardín", "floresta", "edén", "alameda", "arboleda", "reserva
natural"};
        String[] Acantilado = {"acantilado", "Rápido", "abismo", "salto", "precipicio","Isla"};
       // fin de Marcador Natural


       for (int i = 0; i < tamaño; i++) {

          String lema = "", lema1 = "", lema2 = "", lema3 = "", a = "",
          pos = "", pos2 = "", pos3 = "", pos4 = "", posmarc = "", marc = "", posmarca = "";
          //evita desbordamientos en el array
          if (i < tamaño - 1) {
             pos = t.get(1).get(i + 1).toString();
             marc = t.get(0).get(i + 1).toString();
             posmarc = t.get(1).get(i + 1).toString();
             a = t.get(0).get(i + 1).toString();
          }

          if (i < tamaño - 3) {
             pos2 = t.get(1).get(i + 2).toString();
             pos3 = t.get(1).get(i + 3).toString();

             lema1 = t.get(0).get(i + 2).toString();
             lema2 = t.get(0).get(i + 3).toString();
```

```
        }
        if (i < tamaño - 4) {
           pos4 = t.get(1).get(i + 4).toString();
           lema3 = t.get(0).get(i + 4).toString();
        }

        lema = t.get(0).get(i).toString();
        posmarca = t.get(1).get(i).toString();

        //Regla 1      P1 ? (en) NP
        if (lema.equals("en")
              && pos.equalsIgnoreCase("NP")) {
           enviar= t.get(0).get(i + 1).toString();
           extraer += lema + " " + t.get(0).get(i + 1).toString() + "\n";
           in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
           in.createDataPropertyAssertion(ont, iri, "ciudad"+b, prop, enviar);
           b++;

        }

        //Regla numero 2 P2 ? (en) *{1-2} NP
        if (lema.equals("en")) {
           if (pos2.equalsIgnoreCase("NP")) {
                enviar=lema1;
              extraer += lema + " " + lema1 + "\n";
              in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
              in.createDataPropertyAssertion(ont, iri, "ciudad"+b, prop, enviar);
              b++;
           };

if (pos3.equalsIgnoreCase("NP")) {
              enviar="ciudad de "+lema2;
              extraer += lema + " " + lema2 + "\n";
              in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
              in.createDataPropertyAssertion(ont, iri, "ciudad"+b, prop,enviar);
              b++;
           }
        }

        /*Regla numero 3 P3 ?  (en|por|sobre) [MarcadorLingUrbano] * NP
         MarcadorLingUrbano no tiene que ser NP porque sería un apellido. Ej. del Bosque*/
```

```
 // Espacio Urbano Ciudad
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Ciudad.length; j++) {
      if (marc.equalsIgnoreCase(Ciudad[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc+" "+lema1;
            in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
            in.createDataPropertyAssertion(ont, iri, "ciudad"+b, prop,enviar);
            b++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc+" "+lema2;
            in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
            in.createDataPropertyAssertion(ont, iri, "ciudad"+b, prop,enviar);
            b++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc+" "+lema3;
            in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
            in.createDataPropertyAssertion(ont, iri, "ciudad"+b, prop,enviar);
            b++;
         }

      }
   }

}

     // Espacio Urbano Plaza
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre")) ) {

   for (int j = 0; j < Plaza.length; j++) {
      if (marc.equalsIgnoreCase(Plaza[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            enviar= marc + " " + lema1;
```

```
            extraer += lema + " " + marc + " " + lema1 + "\n";
            in.createClassIndividual(ont, iri,"Plaza", "plaza"+c);
            in.createDataPropertyAssertion(ont, iri, "plaza"+c, prop,enviar);
            c++;
          }

          if (pos3.equals("NP") || pos3.equals("NC")) {
            enviar= marc + " " + lema2 ;
            extraer += lema + " " + marc + " " + lema2 + "\n";
            in.createClassIndividual(ont, iri,"Plaza", "plaza"+c);
            in.createDataPropertyAssertion(ont, iri, "plaza"+c, prop,enviar);
            c++;
          }
          if (pos4.equals("NP") || pos4.equals("NC")) {
            enviar= marc + " " + lema3;
            extraer += lema + " " + marc + " " + lema3 + "\n";
            in.createClassIndividual(ont, iri,"Plaza","plaza"+c);
            in.createDataPropertyAssertion(ont, iri, "plaza"+c, prop,enviar);
            c++;
          }

        }
      }

}


// Espacio Urbano Escuela
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Escuela.length; j++) {
      if (marc.equalsIgnoreCase(Escuela[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar= marc + " " + lema1;
            in.createClassIndividual(ont, iri,"Escuela", "escuela"+d);
            in.createDataPropertyAssertion(ont, iri, "escuela"+d, prop,enviar);

            d++;
          }

        if (pos3.equals("NP") || pos3.equals("NC")) {
```

```
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc + " " + lema2;
            in.createClassIndividual(ont, iri,"Escuela", "escuela"+d);
            in.createDataPropertyAssertion(ont, iri, "escuela"+d, prop,enviar);
            d++;
          }
          if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc + " " + lema3;
            in.createClassIndividual(ont, iri,"Escuela", "escuela"+d);
            in.createDataPropertyAssertion(ont, iri, "escuela"+d, prop,enviar);
            d++;
          }

        }
      }

}

// Espacio Urbano Edificio
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Edificio.length; j++) {
      if (marc.equalsIgnoreCase(Edificio[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc + " " + lema1;
            in.createClassIndividual(ont, iri,"Edificio", "edificio"+e);
            in.createDataPropertyAssertion(ont, iri, "edificio"+e, prop,enviar);
            e++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc + " " + lema2;
            in.createClassIndividual(ont, iri,"Edificio", "edificio"+e);
            in.createDataPropertyAssertion(ont, iri, "edificio"+e, prop,enviar);
            e++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc + " " + lema3;
```

```
            in.createClassIndividual(ont, iri,"Edificio", "edificio"+e);
            in.createDataPropertyAssertion(ont, iri, "edificio"+e, prop,enviar);
            e++;
          }

      }
   }

}

// Espacio Urbano localidad
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Localidad.length; j++) {
      if (marc.equalsIgnoreCase(Localidad[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar= marc + " " + lema1;
            in.createClassIndividual(ont, iri,"Localidad", "localidad"+f);
            in.createDataPropertyAssertion(ont, iri, "localidad"+f, prop,enviar);
            f++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar= marc + " " + lema2;
            in.createClassIndividual(ont, iri,"Localidad", "localidad"+f);
            in.createDataPropertyAssertion(ont, iri, "localidad"+f, prop,enviar);
            f++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar= marc + " " + lema3;
            in.createClassIndividual(ont, iri,"Localidad", "localidad"+f);
            in.createDataPropertyAssertion(ont, iri, "localidad"+f, prop,enviar);
            f++;
         }

      }
   }

}
```

```
// Espacio Urbano Colonia
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Colonia.length; j++) {
      if (marc.equalsIgnoreCase(Colonia[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar= marc + " " + lema1;
            in.createClassIndividual(ont, iri,"Colonia", "colonia"+g);
            in.createDataPropertyAssertion(ont, iri, "colonia"+g, prop,enviar);
            g++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar= marc + " " + lema2;
            in.createClassIndividual(ont, iri,"Colonia", "colonia"+g);
            in.createDataPropertyAssertion(ont, iri, "colonia"+g, prop,enviar);
            g++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar= marc + " " + lema3;
            in.createClassIndividual(ont, iri,"Colonia", "colonia"+g);
            in.createDataPropertyAssertion(ont, iri, "colonia"+g, prop,enviar);
            g++;
         }

      }
   }

}

// Espacio Urbano Mercado
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Mercado.length; j++) {
      if (marc.equalsIgnoreCase(Mercado[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar= marc + " " + lema1;
```

```
            in.createClassIndividual(ont, iri,"Mercado", "mercado"+h);
            in.createDataPropertyAssertion(ont, iri, "mercado"+h, prop,enviar);
            h++;
          }

          if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar= marc + " " + lema2;
            in.createClassIndividual(ont, iri,"Mercado", "mercado"+h);
            in.createDataPropertyAssertion(ont, iri, "mercado"+h, prop,enviar);
            h++;
          }
          if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar= marc + " " + lema3;
            in.createClassIndividual(ont, iri,"Mercado", "mercado"+h);
            in.createDataPropertyAssertion(ont, iri, "mercado"+h, prop,enviar);
            h++;
          }

        }
      }

}

// Espacio Urbano Estado
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Estado.length; j++) {
      if (marc.equalsIgnoreCase(Estado[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc + " " + lema1;
            in.createClassIndividual(ont, iri,"Estado", "estado"+k);
            in.createDataPropertyAssertion(ont, iri, "estado"+k, prop,enviar);
            k++;
          }

          if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc + " " + lema2;
            in.createClassIndividual(ont, iri,"Estado", "estado"+k);
```

```
            in.createDataPropertyAssertion(ont, iri, "estado"+k, prop,enviar);
            k++;
          }
          if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc + " " + lema3;
            in.createClassIndividual(ont, iri,"Estado", "estado"+k);
            in.createDataPropertyAssertion(ont, iri, "estado"+k, prop,enviar);
            k++;
          }

        }
      }

}

// Espacio Urbano País    ,
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Pais.length; j++) {
      if (marc.equalsIgnoreCase(Pais[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc + " " + lema1;
            in.createClassIndividual(ont, iri,"Pais", "pais"+l);
            in.createDataPropertyAssertion(ont, iri, "pais"+l, prop,enviar);
            l++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc + " " + lema2;
            in.createClassIndividual(ont, iri,"Pais","pais"+l);
            in.createDataPropertyAssertion(ont, iri, "pais"+l, prop,enviar);
            l++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc + " " + lema3;
            in.createClassIndividual(ont, iri,"Pais", "pais"+l);
            in.createDataPropertyAssertion(ont, iri, "pais"+l, prop,enviar);
            l++;
```

```
        }

      }
    }

}

// Espacio Urbano Calle
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Calle.length; j++) {
      if (marc.equalsIgnoreCase(Calle[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc+" "+lema1;
            in.createClassIndividual(ont, iri,"Calle", "calle"+m);
            in.createDataPropertyAssertion(ont, iri, "calle"+m, prop,enviar);
            m++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc+" "+lema2;
            in.createClassIndividual(ont, iri,"Calle","calle"+m);
            in.createDataPropertyAssertion(ont, iri, "calle"+m, prop,enviar);
            m++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc+" "+lema3;
            in.createClassIndividual(ont, iri,"Calle","calle"+m);
            in.createDataPropertyAssertion(ont, iri, "calle"+m, prop,enviar);
            m++;
         }

      }
   }

}
```

```
//  Regla 4 ?  (en|por|sobre) [MarcadorLingNatural] * NP

   //Espacio Natural acantilado
 if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
        || lema.equalsIgnoreCase("sobre")) ) {
    for (int j = 0; j < Acantilado.length; j++) {
       if (marc.equalsIgnoreCase(Acantilado[j])) {
          if (pos2.equals("NP") || pos2.equals("NC")) {

             extraer += lema + " " + marc + " " + lema1 + "\n";
             enviar=marc+" "+lema1;
             in.createClassIndividual(ont, iri,"Acantilado", "acantilado"+n);
             in.createDataPropertyAssertion(ont, iri, "acantilado"+n, prop,enviar);
             n++;
          }

          if (pos3.equals("NP") || pos3.equals("NC")) {
             extraer += lema + " " + marc + " " + lema2 + "\n";
              enviar=marc+" "+lema2;
             in.createClassIndividual(ont, iri,"Acantilado",  "acantilado"+n);
             in.createDataPropertyAssertion(ont, iri, "acantilado"+n, prop,enviar);
             n++;
          }
          if (pos4.equals("NP") || pos4.equals("NC")) {
             extraer += lema + " " + marc + " " + lema3 + "\n";
              enviar=marc+" "+lema3;
             in.createClassIndividual(ont, iri,"Acantilado",  "acantilado"+n);
             in.createDataPropertyAssertion(ont, iri, "acantilado"+n, prop,enviar);
             n++;
          }

       }
    }

 }

 //Espacio Natural Bosque
 if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
        || lema.equalsIgnoreCase("sobre")) ) {
    for (int j = 0; j < Bosque.length; j++) {
       if (marc.equalsIgnoreCase(Bosque[j])) {
          if (pos2.equals("NP") || pos2.equals("NC")) {
```

```
                extraer += lema + " " + marc + " " + lema1 + "\n";
                 enviar=marc+" "+lema1;
                in.createClassIndividual(ont, iri,"Bosque", "boque"+o);
                in.createDataPropertyAssertion(ont, iri, "boque"+o, prop,enviar);
                o++;
              }

              if (pos3.equals("NP") || pos3.equals("NC")) {
                extraer += lema + " " + marc + " " + lema2 + "\n";
                 enviar=marc+" "+lema2;
                in.createClassIndividual(ont, iri,"Bosque", "boque"+o);
                in.createDataPropertyAssertion(ont, iri, "boque"+o, prop,enviar);
                o++;
              }
              if (pos4.equals("NP") || pos4.equals("NC")) {
                extraer += lema + " " + marc + " " + lema3 + "\n";
                 enviar=marc+" "+lema3;
                in.createClassIndividual(ont, iri,"Bosque", "boque"+o);
                in.createDataPropertyAssertion(ont, iri, "boque"+o, prop,enviar);
                o++;
              }

            }
          }

        }
//Espacio Natural Montaña
        if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
            || lema.equalsIgnoreCase("sobre"))) {
          for (int j = 0; j < Montaña.length; j++) {
            if (marc.equalsIgnoreCase(Montaña[j])) {
              if (pos2.equals("NP") || pos2.equals("NC")) {
                extraer += lema + " " + marc + " " + lema1 + "\n";
                 enviar=marc+" "+lema1;
                in.createClassIndividual(ont, iri,"Montaña", "montaña"+p);
                in.createDataPropertyAssertion(ont, iri,"montaña"+p, prop,enviar);
                p++;
              }

              if (pos3.equals("NP") || pos3.equals("NC")) {
                extraer += lema + " " + marc + " " + lema2 + "\n";
                enviar=marc+" "+lema2;
```

```
            in.createClassIndividual(ont, iri,"Montaña", "montaña"+p);
            in.createDataPropertyAssertion(ont, iri,"montaña"+p, prop,enviar);
            p++;
          }
          if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc+" "+lema3;
            in.createClassIndividual(ont, iri,"Montaña", "montaña"+p);
            in.createDataPropertyAssertion(ont, iri,"montaña"+p, prop,enviar);
            p++;
          }

        }
      }

}

//Espacio Natural Sierra
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre")) ) {
   for (int j = 0; j < Sierra.length; j++) {
      if (marc.equalsIgnoreCase(Sierra[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc+" "+lema1;
            in.createClassIndividual(ont, iri,"Sierra", "sierra"+q);
            in.createDataPropertyAssertion(ont, iri,"sierra"+q, prop,enviar);
            q++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc+" "+lema2;
            in.createClassIndividual(ont, iri,"Sierra", "sierra"+q);
            in.createDataPropertyAssertion(ont, iri,"sierra"+q, prop,enviar);
            q++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc+" "+lema3;
            in.createClassIndividual(ont, iri,"Sierra", "sierra"+q);
            in.createDataPropertyAssertion(ont, iri,"sierra"+q, prop,enviar);
```

```
            q++;
          }

      }
    }

}

//Espacio Natural Lago
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Lago.length; j++) {
      if (marc.equalsIgnoreCase(Lago[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc+" "+lema1;
            in.createClassIndividual(ont, iri,"Lago", "lago"+r);
            in.createDataPropertyAssertion(ont, iri,"lago"+r, prop,enviar);
            r++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc+" "+lema2;
            in.createClassIndividual(ont, iri,"Lago", "lago"+r);
             in.createDataPropertyAssertion(ont, iri,"lago"+r, prop,enviar);
             r++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc+" "+lema3;
            in.createClassIndividual(ont, iri,"Lago", "lago"+r);
             in.createDataPropertyAssertion(ont, iri,"lago"+r, prop,enviar);
             r++;
         }

      }
   }

}

//Espacio Natural Rio
```

```
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre")) ) {
   for (int j = 0; j < Rio.length; j++) {
      if (marc.equalsIgnoreCase(Rio[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc+" "+lema1;
            in.createClassIndividual(ont, iri,"Rio", "rio"+s);
            in.createDataPropertyAssertion(ont, iri,"rio"+s, prop,enviar);
            s++;
         }

         if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc+" "+lema2;
            in.createClassIndividual(ont, iri,"Rio", "rio"+s);
            in.createDataPropertyAssertion(ont, iri,"rio"+s, prop,enviar);
            s++;
         }
         if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc+" "+lema3;
            in.createClassIndividual(ont, iri,"Rio","rio"+s);
            in.createDataPropertyAssertion(ont, iri,"rio"+s, prop,enviar);
            s++;
         }

      }
   }

}

//Espacio Natural Parque
if ((lema.equalsIgnoreCase("en") || lema.equalsIgnoreCase("por")
      || lema.equalsIgnoreCase("sobre"))) {
   for (int j = 0; j < Parque.length; j++) {
      if (marc.equalsIgnoreCase(Parque[j])) {
         if (pos2.equals("NP") || pos2.equals("NC")) {
            extraer += lema + " " + marc + " " + lema1 + "\n";
            enviar=marc+" "+lema1;
            in.createClassIndividual(ont, iri,"Parque", "parque"+u);
            in.createDataPropertyAssertion(ont, iri,"parque"+u, prop,enviar);
```

```
            u++;
          }

          if (pos3.equals("NP") || pos3.equals("NC")) {
            extraer += lema + " " + marc + " " + lema2 + "\n";
            enviar=marc+" "+lema2;
            in.createClassIndividual(ont, iri,"Parque", "parque"+u);
            in.createDataPropertyAssertion(ont, iri,"parque"+u, prop,enviar);
            u++;
          }
          if (pos4.equals("NP") || pos4.equals("NC")) {
            extraer += lema + " " + marc + " " + lema3 + "\n";
            enviar=marc+" "+lema3;
            in.createClassIndividual(ont, iri,"Parque", "parque"+u);
            in.createDataPropertyAssertion(ont, iri,"parque"+u, prop,enviar);
            u++;
          }

        }
      }

}

//Regla 5      P5 ?   [MarcadorLingUrbano] *{1-2} (NP or NC)
// Espacio Urbano Plaza
for (int j = 0; j < Plaza.length; j++) {
   if (lema.equalsIgnoreCase(Plaza[j]) ) {
      //duda * vale 0
      if (pos.equals("NP")) {
         extraer += lema + " " + a + "\n";
         enviar=lema+" "+a;
         in.createClassIndividual(ont, iri,"Plaza", "plaza"+c);
         in.createDataPropertyAssertion(ont, iri,"plaza"+c, prop,enviar);
         c++;
      }

      if (pos2.equals("NP")) {
         extraer += lema + " " + lema1 + "\n";
         enviar=lema+" "+lema1;
         in.createClassIndividual(ont, iri,"Plaza","plaza"+c);
         in.createDataPropertyAssertion(ont, iri,"plaza"+c, prop,enviar);
         c++;
```

```
            }
            if (pos3.equals("NP")) {
               extraer += lema + " " + lema2 + "\n";
               enviar=lema+" "+lema2;
               in.createClassIndividual(ont, iri,"Plaza","plaza"+c);
               in.createDataPropertyAssertion(ont, iri,"plaza"+c, prop,enviar);
               c++;
            }

         }
      }

      // Espacio Urbano Ciudad
      for (int j = 0; j < Ciudad.length; j++) {
         if (lema.equalsIgnoreCase(Ciudad[j]) ) {
            //duda * vale 0
            if (pos.equals("NP")) {
               extraer += lema + " " + a + "\n";
               enviar=lema+" "+a;
               in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
               in.createDataPropertyAssertion(ont, iri,"ciudad"+b, prop,enviar);
               b++;
            }

            if (pos2.equals("NP")) {
               extraer += lema + " " + lema1 + "\n";
               enviar=lema+" "+lema1;
               in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
               in.createDataPropertyAssertion(ont, iri,"ciudad"+b, prop,enviar);
               b++;
            }
            if (pos3.equals("NP")) {
               extraer += lema + " " + lema2 + "\n";
               enviar=lema+" "+lema2;
               in.createClassIndividual(ont, iri,"Ciudad", "ciudad"+b);
               in.createDataPropertyAssertion(ont, iri,"ciudad"+b, prop,enviar);
               b++;
            }

         }
      }
// Espacio Urbano Escuela
```

```
for (int j = 0; j < Escuela.length; j++) {
   if (lema.equalsIgnoreCase(Escuela[j]) ) {
      //duda * vale 0
      if (pos.equals("NP")) {
         extraer += lema + " " + a + "\n";
         enviar=lema+" "+a;
         in.createClassIndividual(ont, iri,"Escuela", "escuela"+d);
         in.createDataPropertyAssertion(ont, iri,"escuela"+d,prop,enviar);
         d++;
      }

      if (pos2.equals("NP")) {
         extraer += lema + " " + lema1 + "\n";
         enviar=lema+" "+lema1;
         in.createClassIndividual(ont, iri,"Escuela", "escuela"+d);
         in.createDataPropertyAssertion(ont, iri,"escuela"+d,prop,enviar);
         d++;
      }
      if (pos3.equals("NP")) {
         extraer += lema + " " + lema2 + "\n";
         enviar=lema+" "+lema2;
         in.createClassIndividual(ont, iri,"Escuela","escuela"+d);
         in.createDataPropertyAssertion(ont, iri,"escuela"+d,prop,enviar);
         d++;
      }

   }
}

// Espacio Urbano Edificio
for (int j = 0; j < Edificio.length; j++) {
   if (lema.equalsIgnoreCase(Edificio[j]) ) {
      //duda * vale 0
      if (pos.equals("NP")) {
         extraer += lema + " " + a + "\n";
         enviar=lema+" "+a;
         in.createClassIndividual(ont, iri,"Edificio", "edificio"+e);
         in.createDataPropertyAssertion(ont, iri,"edificio"+e,prop,enviar);
         e++;
      }

      if (pos2.equals("NP")) {
```

```
            extraer += lema + " " + lema1 + "\n";
             enviar=lema+" "+lema1;
            in.createClassIndividual(ont, iri,"Edificio", "edificio"+e);
            in.createDataPropertyAssertion(ont, iri,"edificio"+e,prop,enviar);
            e++;
         }
         if (pos3.equals("NP")) {
            extraer += lema + " " + lema2 + "\n";
             enviar=lema+" "+lema2;
            in.createClassIndividual(ont, iri,"Edificio", "edificio"+e);
            in.createDataPropertyAssertion(ont, iri,"edificio"+e,prop,enviar);
            e++;
         }

      }
   }
   // Espacio Urbano localidad
   for (int j = 0; j < Localidad.length; j++) {
      if (lema.equalsIgnoreCase(Localidad[j])) {
         //duda * vale 0
         if (pos.equals("NP")) {
            extraer += lema + " " + a + "\n";
             enviar=lema+" "+a;
            in.createClassIndividual(ont, iri,"Localidad", "localidad"+f);
            in.createDataPropertyAssertion(ont, iri,"localidad"+f,prop,enviar);
            f++;
         }

         if (pos2.equals("NP")) {
            extraer += lema + " " + lema1 + "\n";
             enviar=lema+" "+lema1;
            in.createClassIndividual(ont, iri,"Localidad","localidad"+f);
            in.createDataPropertyAssertion(ont, iri,"localidad"+f,prop,enviar);
            f++;
         }
         if (pos3.equals("NP")) {
            extraer += lema + " " + lema2 + "\n";
             enviar=lema+" "+lema2;
            in.createClassIndividual(ont, iri,"Localidad", "localidad"+f);
            in.createDataPropertyAssertion(ont, iri,"localidad"+f,prop,enviar);
            f++;
         }
```

```
   }
}

  // Espacio Urbano Colonia
for (int j = 0; j < Colonia.length; j++) {
   if (lema.equalsIgnoreCase(Colonia[j])) {
      //duda * vale 0
      if (pos.equals("NP")) {
         extraer += lema + " " + a + "\n";
          enviar=lema+" "+a;
         in.createClassIndividual(ont, iri,"Colonia", "colonia"+g);
         in.createDataPropertyAssertion(ont, iri, "colonia"+g,prop,enviar);
         g++;
      }

      if (pos2.equals("NP")) {
         extraer += lema + " " + lema1 + "\n";
          enviar=lema+" "+lema1;
         in.createClassIndividual(ont, iri,"Colonia", "colonia"+g);
         in.createDataPropertyAssertion(ont, iri, "colonia"+g,prop,enviar);
           g++;
      }
      if (pos3.equals("NP")) {
         extraer += lema + " " + lema2 + "\n";
          enviar=lema+" "+lema2;
         in.createClassIndividual(ont, iri,"Colonia", "colonia"+g);
         in.createDataPropertyAssertion(ont, iri, "colonia"+g,prop,enviar);
           g++;
      }

   }
}

// Espacio Urbano Mercado
for (int j = 0; j < Mercado.length; j++) {
   if (lema.equalsIgnoreCase(Mercado[j]) ) {
      //duda * vale 0
      if (pos.equals("NP")) {
         extraer += lema + " " + a + "\n";
          enviar=lema+" "+a;
         in.createClassIndividual(ont, iri,"Mercado", "mercado"+h);
```

```
            in.createDataPropertyAssertion(ont, iri,"mercado"+h,prop,enviar);
            h++;
        }

        if (pos2.equals("NP")) {
            extraer += lema + " " + lema1 + "\n";
            enviar=lema+" "+lema1;
            in.createClassIndividual(ont, iri,"Mercado", "mercado"+h);
            in.createDataPropertyAssertion(ont, iri,"mercado"+h,prop,enviar);
             h++;
        }
        if (pos3.equals("NP")) {
            extraer += lema + " " + lema2 + "\n";
            enviar=lema+" "+lema2;
            in.createClassIndividual(ont, iri,"Mercado", "mercado"+h);
            in.createDataPropertyAssertion(ont, iri,"mercado"+h,prop,enviar);
             h++;
        }

    }
}

// Espacio Urbano Estado
for (int j = 0; j < Estado.length; j++) {
    if (lema.equalsIgnoreCase(Estado[j])) {
        //duda * vale 0
        if (pos.equals("NP")) {
            extraer += lema + " " + a + "\n";
            enviar=lema+" "+a;
            in.createClassIndividual(ont, iri,"Estado","estado"+k);
            in.createDataPropertyAssertion(ont, iri,"estado"+k,prop,enviar);
            k++;
        }

        if (pos2.equals("NP")) {
            extraer += lema + " " + lema1 + "\n";
            enviar=lema+" "+lema1;
            in.createClassIndividual(ont, iri,"Estado", "estado"+k);
            in.createDataPropertyAssertion(ont, iri,"estado"+k,prop,enviar);
            k++;
        }
        if (pos3.equals("NP")) {
```

```
            extraer += lema + " " + lema2 + "\n";
            enviar=lema+" "+lema2;
            in.createClassIndividual(ont, iri,"Estado", "estado"+k);
            in.createDataPropertyAssertion(ont, iri,"estado"+k,prop,enviar);
             k++;
        }

    }
}

// Espacio Urbano País    ,
for (int j = 0; j < Pais.length; j++) {
    if (lema.equalsIgnoreCase(Pais[j])) {
        //duda * vale 0
        if (pos.equals("NP")) {
            extraer += lema + " " + a + "\n";
            enviar=lema+" "+a;
            in.createClassIndividual(ont, iri,"Pais","pais"+l);
            in.createDataPropertyAssertion(ont, iri,"pais"+l,prop,enviar);
            l++;
        }

        if (pos2.equals("NP")) {
            extraer += lema + " " + lema1 + "\n";
             enviar=lema+" "+lema1;
            in.createClassIndividual(ont, iri,"Pais", "pais"+l);
            in.createDataPropertyAssertion(ont, iri,"pais"+l,prop,enviar);
            l++;

        }
        if (pos3.equals("NP")) {
            extraer += lema + " " + lema2 + "\n";
             enviar=lema+" "+lema2;
            in.createClassIndividual(ont, iri,"Pais", "pais"+l);
            in.createDataPropertyAssertion(ont, iri,"pais"+l,prop,enviar);
            l++;
        }

    }
}
// Espacio Urbano Calle
for (int j = 0; j < Calle.length; j++) {
```

```
   if (lema.equalsIgnoreCase(Calle[j])) {
      //duda * vale 0
      if (pos.equals("NP")) {
         extraer += lema + " " + a + "\n";
          enviar=lema+" "+a;
         in.createClassIndividual(ont, iri,"Calle","calle"+m);
         in.createDataPropertyAssertion(ont, iri,"calle"+m,prop,enviar);
         m++;
      }

      if (pos2.equals("NP")) {
         extraer += lema + " " + lema1 + "\n";
          enviar=lema+" "+lema1;
         in.createClassIndividual(ont, iri,"Calle", "calle"+m);
         in.createDataPropertyAssertion(ont, iri,"calle"+m,prop,enviar);
         m++;
      }
      if (pos3.equals("NP")) {
         extraer += lema + " " + lema2 + "\n";
          enviar=lema+" "+lema2;
         in.createClassIndividual(ont, iri,"Calle","calle"+m);
         in.createDataPropertyAssertion(ont, iri,"calle"+m,prop,enviar);
         m++;
      }

   }
}



//Regla 6      P6 ?   [MarcadorLingNatural] *{1-2} (NP or NC)

//Espacio Natural acantilado
for (int j = 0; j < Acantilado.length; j++) {
   if (lema.equalsIgnoreCase(Acantilado[j])) {
      //duda * vale 0
      if (pos.equals("NP")) {
         extraer += lema + " " + a + "\n";
          enviar=lema+" "+a;
         in.createClassIndividual(ont, iri,"Acantilado","acantilado"+n);
         in.createDataPropertyAssertion(ont, iri,"acantilado"+n,prop,enviar);
         n++;
```

```
        }

        if (pos2.equals("NP")) {
           extraer += lema + " " + lema1 + "\n";
            enviar=lema+" "+lema1;
           in.createClassIndividual(ont, iri,"Acantilado", "acantilado"+n);
           in.createDataPropertyAssertion(ont, iri,"acantilado"+n,prop,enviar);
            n++;
        }
        if (pos3.equals("NP")) {
           extraer += lema + " " + lema2 + "\n";
            enviar=lema+" "+lema2;
           in.createClassIndividual(ont, iri,"Acantilado","acantilado"+n);
           in.createDataPropertyAssertion(ont, iri,"acantilado"+n,prop,enviar);
            n++;
        }

    }
}

//Espacio Natural Bosque
for (int j = 0; j < Bosque.length; j++) {
    if (lema.equalsIgnoreCase(Bosque[j]) ) {
        //duda * vale 0
        if (pos.equals("NP")) {
           extraer += lema + " " + a + "\n";
            enviar=lema+" "+a;
           in.createClassIndividual(ont, iri,"Bosque","bosque"+o);
           in.createDataPropertyAssertion(ont, iri,"bosque"+o,prop,enviar);
           o++;
        }

        if (pos2.equals("NP")) {
           extraer += lema + " " + lema1 + "\n";
            enviar=lema+" "+lema1;
           in.createClassIndividual(ont, iri,"Bosque","bosque"+o);
           in.createDataPropertyAssertion(ont, iri,"bosque"+o,prop,enviar);
            o++;
        }
        if (pos3.equals("NP")) {
           extraer += lema + " " + lema2 + "\n";
            enviar=lema+" "+lema2;
```

```
            in.createClassIndividual(ont, iri,"Bosque","bosque"+o);
            in.createDataPropertyAssertion(ont, iri,"bosque"+o,prop,enviar);
             o++;
        }

    }
}

//Espacio Natural Montaña
for (int j = 0; j < Montaña.length; j++) {
    if (lema.equalsIgnoreCase(Montaña[j])) {
        //duda * vale 0
        if (pos.equals("NP")) {
            extraer += lema + " " + a + "\n";
             enviar=lema+" "+a;
            in.createClassIndividual(ont, iri,"Montaña","montaña"+p);
            in.createDataPropertyAssertion(ont, iri,"montaña"+p,prop,enviar);
            p++;
        }

        if (pos2.equals("NP")) {
            extraer += lema + " " + lema1 + "\n";
            enviar=lema+" "+lema1;
            in.createClassIndividual(ont, iri,"Montaña","montaña"+p);
            in.createDataPropertyAssertion(ont, iri,"montaña"+p,prop,enviar);
            p++;
        }
        if (pos3.equals("NP")) {
            extraer += lema + " " + lema2 + "\n";
            enviar=lema+" "+lema2;
            in.createClassIndividual(ont, iri,"Montaña","montaña"+p);
            in.createDataPropertyAssertion(ont, iri,"montaña"+p,prop,enviar);
            p++;
        }

    }
}
//Espacio Natural Sierra
for (int j = 0; j < Sierra.length; j++) {
    if (lema.equalsIgnoreCase(Sierra[j]) ) {
        //duda * vale 0
        if (pos.equals("NP")) {
```

```
            extraer += lema + " " + a + "\n";
            enviar=lema+" "+a;
            in.createClassIndividual(ont, iri,"Sierra","sierra"+q);
            in.createDataPropertyAssertion(ont, iri,"sierra"+q,prop,enviar);
            q++;
        }

        if (pos2.equals("NP")) {
            extraer += lema + " " + lema1 + "\n";
             enviar=lema+" "+lema1;
            in.createClassIndividual(ont, iri,"Sierra","sierra"+q);
            in.createDataPropertyAssertion(ont, iri,"sierra"+q,prop,enviar);
            q++;
        }
        if (pos3.equals("NP")) {
            extraer += lema + " " + lema2 + "\n";
             enviar=lema+" "+lema2;
            in.createClassIndividual(ont, iri,"Sierra","sierra"+q);
            in.createDataPropertyAssertion(ont, iri,"sierra"+q,prop,enviar);
            q++;
        }

    }
}
//Espacio Natural Lago
for (int j = 0; j < Lago.length; j++) {
    if (lema.equalsIgnoreCase(Lago[j])) {
        //duda * vale 0
        if (pos.equals("NP")) {
            extraer += lema + " " + a + "\n";
             enviar=lema+" "+a;
            in.createClassIndividual(ont, iri,"Lago","lago"+r);
            in.createDataPropertyAssertion(ont, iri,"lago"+r,prop,enviar);
            r++;
        }

        if (pos2.equals("NP")) {
            extraer += lema + " " + lema1 + "\n";
             enviar=lema+" "+lema1;
            in.createClassIndividual(ont, iri,"Lago","lago"+r);
            in.createDataPropertyAssertion(ont, iri,"lago"+r,prop,enviar);
            r++;
```

```
        }
        if (pos3.equals("NP")) {
           extraer += lema + " " + lema2 + "\n";
            enviar=lema+" "+lema2;
           in.createClassIndividual(ont, iri,"Lago","lago"+r);
           in.createDataPropertyAssertion(ont, iri,"lago"+r,prop,enviar);
           r++;
        }

     }
}

//Espacio Natural Rio
for (int j = 0; j < Rio.length; j++) {
   if (lema.equalsIgnoreCase(Rio[j])) {
      //duda * vale 0
      if (pos.equals("NP")) {
         extraer += lema + " " + a + "\n";
          enviar=lema+" "+a;
         in.createClassIndividual(ont, iri,"Rio","rio"+s);
         in.createDataPropertyAssertion(ont, iri,"rio"+s,prop,enviar);
         s++;
      }

      if (pos2.equals("NP")) {
         extraer += lema + " " + lema1 + "\n";
          enviar=lema+" "+lema1;
         in.createClassIndividual(ont, iri,"Rio","rio"+s);
         in.createDataPropertyAssertion(ont, iri,"rio"+s,prop,enviar);
         s++;
      }
      if (pos3.equals("NP")) {
         extraer += lema + " " + lema2 + "\n";
          enviar=lema+" "+lema2;
         in.createClassIndividual(ont, iri,"Rio","rio"+s);
         in.createDataPropertyAssertion(ont, iri,"rio"+s,prop,enviar);
         s++;
      }

   }
}
```

```
        //Espacio Natural Parque
        for (int j = 0; j < Parque.length; j++) {
           if (lema.equalsIgnoreCase(Parque[j]) ) {
              //duda * vale 0
              if (pos.equals("NP")) {
                 extraer += lema + " " + a + "\n";
                  enviar=lema+" "+a;
                 in.createClassIndividual(ont, iri,"Parque","parque"+u);
                 in.createDataPropertyAssertion(ont, iri,"parque"+u,prop,enviar);
                 u++;
              }

              if (pos2.equals("NP")) {
                 extraer += lema + " " + lema1 + "\n";
                  enviar=lema+" "+lema1;
                 in.createClassIndividual(ont, iri,"Parque","parque"+u);
                 in.createDataPropertyAssertion(ont, iri,"parque"+u,prop,enviar);
                 u++;

              }
              if (pos3.equals("NP")) {
                 extraer += lema + " " + lema2 + "\n";
                  enviar=lema+" "+lema2;
                 in.createClassIndividual(ont, iri,"Parque","parque"+u);
                 in.createDataPropertyAssertion(ont, iri,"parque"+u,prop,enviar);
                 u++;
              }

           }
        }

     }
     return extraer;
  }

}
```

10.5 Anexo C: Modelo de datos semántico Espacio

```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.semanticweb.org/5/ontologies/2014/9/Espacio"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    ontologyIRI="http://www.semanticweb.org/5/ontologies/2014/9/Espacio">
   <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Declaration>
      <Class IRI="#Acantilado"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Bosque"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Calle"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Ciudad"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Colonia"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Edificio"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Escuela"/>
   </Declaration>
```

```
<Declaration>
   <Class IRI="#Espacio"/>
</Declaration>
<Declaration>
   <Class IRI="#EspacioNatural"/>
</Declaration>
<Declaration>
   <Class IRI="#EspacioUrbano"/>
</Declaration>
<Declaration>
   <Class IRI="#Estado"/>
</Declaration>
<Declaration>
   <Class IRI="#Lago"/>
</Declaration>
<Declaration>
   <Class IRI="#Localidad"/>
</Declaration>
<Declaration>
   <Class IRI="#Mercado"/>
</Declaration>
<Declaration>
   <Class IRI="#Montaña"/>
</Declaration>
<Declaration>
   <Class IRI="#Pais"/>
</Declaration>
<Declaration>
   <Class IRI="#Parque"/>
</Declaration>
<Declaration>
   <Class IRI="#Plaza"/>
</Declaration>
<Declaration>
   <Class IRI="#Rio"/>
</Declaration>
<Declaration>
   <Class IRI="#Sierra"/>
</Declaration>
<Declaration>
   <DataProperty IRI="#tieneNombre"/>
</Declaration>
```

```
<SubClassOf>
   <Class IRI="#Acantilado"/>
   <Class IRI="#EspacioNatural"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#Bosque"/>
   <Class IRI="#EspacioNatural"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#Calle"/>
   <Class IRI="#EspacioUrbano"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#Ciudad"/>
   <Class IRI="#EspacioUrbano"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#Colonia"/>
   <Class IRI="#EspacioUrbano"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#Edificio"/>
   <Class IRI="#EspacioUrbano"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#Escuela"/>
   <Class IRI="#EspacioUrbano"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#EspacioNatural"/>
   <Class IRI="#Espacio"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#EspacioUrbano"/>
   <Class IRI="#Espacio"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#Estado"/>
   <Class IRI="#EspacioUrbano"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#Lago"/>
```

```
    <Class IRI="#EspacioNatural"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Localidad"/>
    <Class IRI="#EspacioUrbano"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Mercado"/>
    <Class IRI="#EspacioUrbano"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Montaña"/>
    <Class IRI="#EspacioNatural"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Pais"/>
    <Class IRI="#EspacioUrbano"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Parque"/>
    <Class IRI="#EspacioNatural"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Plaza"/>
    <Class IRI="#EspacioUrbano"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Rio"/>
    <Class IRI="#EspacioNatural"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Sierra"/>
    <Class IRI="#EspacioNatural"/>
  </SubClassOf>
  <DataPropertyDomain>
    <DataProperty IRI="#tieneNombre"/>
    <Class IRI="#Espacio"/>
  </DataPropertyDomain>
  <DataPropertyRange>
    <DataProperty IRI="#tieneNombre"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataPropertyRange> </Ontology>
```

10.6 Código ontología Tiempo.

```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.semanticweb.org/satellite/ontologies/2014/9/Tiempo"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    ontologyIRI="http://www.semanticweb.org/satellite/ontologies/2014/9/Tiempo">
   <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Declaration>
      <Class IRI="#Fecha"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Hora"/>
   </Declaration>
   <Declaration>
      <Class IRI="#HoraExacta"/>
   </Declaration>
   <Declaration>
      <Class IRI="#HoraPeriodo"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Tiempo"/>
   </Declaration>
   <Declaration>
      <DataProperty IRI="#tieneNombre"/>
   </Declaration>
   <SubClassOf>
      <Class IRI="#Fecha"/>
```

```
      <Class IRI="#Tiempo"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#Hora"/>
      <Class IRI="#Tiempo"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#HoraExacta"/>
      <Class IRI="#Hora"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#HoraPeriodo"/>
      <Class IRI="#Hora"/>
   </SubClassOf>
   <DataPropertyDomain>
      <DataProperty IRI="#tieneNombre"/>
      <Class IRI="#Tiempo"/>
   </DataPropertyDomain>
   <DataPropertyRange>
      <DataProperty IRI="#tieneNombre"/>
      <Datatype abbreviatedIRI="xsd:string"/>
   </DataPropertyRange>
</Ontology>


<!-- Generated by the OWL API (version 3.5.0) http://owlapi.sourceforge.net -->
```

11. BIBLIOGRAFÍA

[1] S.M. Ugalde Chávez, "*Sistema de recuperación de información semántico",* proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.

[2] J. L. Ugalde Anaya, "*Sistema clasificador de documentos de proyectos terminales usando el concepto de memoria asociativa",* proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2011.

[3] F. Alejandro Acosta, "*Sistema de procesamiento de textos de investigación"*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2013.

[4] A. Téllez Valera, "*Extracción de información con algoritmos de clasificación"*, tesis de maestría, Instituto Nacional de Astro física Óptica y Electrónica, México, Puebla, 2005.

[5] P. Martínez Fernández, A. García Serrano, *"Utilizando recursos lingüísticos para la mejora de recuperación de información en la web"*, 2002.

[6] Fader A., Schmitz M., Bart R., Soderland S., Etzioni O., ReVerb, "Computer Science & Engineering University of Washington".

[7] Stephen D. Richardson "*Natural language processing the PLN"* approach / edited by Karen Jensen, George E. Heiddorn

[8] C. G. Figuerola. "*La investigación sobre recuperación de información en español".* Editores, Documentación, Terminología y Traducción, pages 73-82. Síntesis, Madrid, 2000.

[9] J.A. Reyes Ortiz., "Creación Automática de Ontologías a partir de textos", tesis de doctorado, Centro Nacional de Investigación y Desarrollo Tecnológico, Morelos, 2013.

[10] T. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In Formal Analysis in Conceptual Analysis and Knowledge Representation. Kluwer, 1993.

[11] H. Weigand. Multilingual ontology-based lexicon for news _ltering. In K. Mahesh, editor, The TREVI Project, pages 138-159, 1997.

[12] TreeTagger desarrollado por Helmut Schmid "Universidad de Stuttgart".