Interfaz de lenguaje natural para deducción de información almacenada en ontologías

Alejandro Solís-Sánchez, Rogelio Florencia-Juárez, Juan Carlos Acosta-Guadarrama, Francisco López-Orozco

Universidad Autónoma de Ciudad Juárez, Juárez, Chihuahua, México

al160509@alumnos.uacj.mx, {rogelio.florencia, juan.acosta, francisco.orozco}@uacj.mx

Resumen. Las interfaces de lenguaje natural a ontologías permiten consultar datos de un dominio específico, almacenados en ontologías diseñadas mediante el uso de lenguajes de ontología, como el lenguaje OWL. Para acceder a estos datos, se requieren lenguajes de consulta complejos como SPARQL, utilizados solo por usuarios especializados. Las interfaces de lenguaje natural a ontologías traducen consultas formuladas en lenguaje natural a consultas SPARQL. En este trabajo, se describe la arquitectura de una interfaz de lenguaje natural a ontologías. Recibe una consulta formulada por los usuarios en un dispositivo Android utilizando su voz. La consulta se transforma en texto y se envía a un servidor Java Server Pages (JSP). El núcleo de nuestra interfaz, que se ejecuta en un servidor JSP, recibe la consulta del usuario convertida en texto y, mediante técnicas de procesamiento de lenguaje natural, la transforma en una consulta SPARQL, que se utiliza para extraer de la ontología la información solicitada por los usuarios. Para traducir consultas en lenguaje natural a consultas SPARQL, la interfaz utiliza técnicas de procesamiento del lenguaje natural como Tokenización, Lematización y etiquetado gramatical, así como la extracción del conocimiento almacenado en la ontología a consultar y diálogos de aclaración para responder las preguntas del usuario.

Palabras clave: Interfaces de lenguaje natural a ontologías, SPARQL, OWL, Web semántica.

Natural Language Interface for Information Deduction Stored in Ontologies

Abstract. Natural language interfaces to ontologies allow querying data of a specific domain, stored in ontologies designed by using ontology languages such as the Ontology Web Language. To access this data, complex query languages such as SPARQL, used by only specialized users, are required. Natural language interfaces to ontologies translate queries formulated in natural language to SPARQL queries. In this paper, the architecture of a natural language interface to ontologies is described. It receives a query formulated by users on an Android device using their voice. The query is transformed into text and it is sent to a Java Server Page (JSP) server. The core of our interface, running on the JSP server, receives the user's query converted to text and, through natural language

processing techniques, it transforms it into a SPARQL query, which is used to extract from the ontology to query, the information requested by users. To translate natural language queries to SPARQL queries, the interface uses natural language processing techniques such as Tokenization, Lemmatization, and Part of Speech tagging, as well as, extraction of knowledge stored in the ontology to be queried, and clarification dialogues to answer user questions.

Keywords: Natural language interfaces to ontologies, SPARQL, OWL, Semantic Web.

1. Introducción

La Web Semántica es una extensión de la web actual [1], donde la información recibe un significado bien definido, que puede ser entendido por computadoras y humanos. La arquitectura de la Web Semántica que fue establecida por Tim Berners-Lee (considerado como el creador de la Web), hace uso de ontologías como uno de sus principales componentes. Una ontología es una especificación explícita de una conceptualización. Las ontologías permiten comprender mejor la estructura del conocimiento, ya que muestran los conceptos y las relaciones que existen entre ellos.

En la Web Semántica la información es almacenada haciendo uso de las ontologías. Dichas ontologías son representadas a través del lenguaje OWL (considerado una extensión del lenguaje *Resource Description Framework* RDF) [2]. Consultar el conocimiento almacenado en una ontología requiere de conocimientos avanzados diseñados para tal propósito, un ejemplo es SPARQL (Protocol and RDF Query Language) [3], conocimientos que la mayoría de las personas no posee [4]. Por tal motivo se han implementado interfaces que reciben una consulta en lenguaje natural y la transforman a una consulta SPARQL con la cual extraen la información solicitada por los usuarios.

La Web Semántica necesita de interfaces para consultar ontologías en un lenguaje más cercano al que los humanos conocemos y entendemos. Esto nos facilita la interacción con sitios o aplicaciones basados en la arquitectura de la misma. En este trabajo se propone la arquitectura para una interfaz que, mediante consultas en lenguaje natural, permita buscar información almacenada en una ontología, utilizando un dispositivo móvil para su funcionamiento. En la sección 2 se agrega una descripción de trabajos relacionados existentes, en la sección 3 se agrega la descripción de la propuesta de este trabajo, la sección 4 describe la implementación y en la sección 5 las conclusiones.

2. Trabajos relacionados

Existen diferentes arquitecturas propuestas para la implementación de interfaces de lenguaje natural a ontologías, utilizan diferentes técnicas de procesamiento de lenguaje natural y lenguajes formales de consulta. A continuación, se describe la arquitectura de algunas interfaces de lenguaje natural a ontologías como son FREYA [5], Gingseng [6], QuestIO [7], ORAKEL [8] y AquaLog [9].

2.1 FREYA

Es una interfaz de lenguaje natural (ILN) que puede usarse para consultar ontologías en formato RDF/OWL. FREyA (Feedback Refinement and Extended Vocabulary Aggregation) mapea a SPARQL una consulta en lenguaje natural en forma de pregunta, luego se ejecuta en un repositorio RDF/OWL y devuelve una respuesta [5]. FREyA usa el conocimiento disponible en la ontología para identificar los términos que se encuentren en la consulta. A dichos términos se les llama (OC). Si existe ambigüedad en los términos, FREyA genera un diálogo de clarificación en el que el usuario elige una de varias opciones. En el diálogo de clarificación el usuario elige una opción que es almacenada y usada para entrenar al sistema.

A partir de los OC FREyA genera un conjunto de tripletas que convierte a consultas SPARQL para buscar la respuesta en la ontología. Para mostrar los resultados al usuario, se identifica primero el tipo de respuesta [5]. Para el tipo de respuesta realiza un análisis sintáctico y una búsqueda en la ontología para identificar el *focus* de la pregunta. Un *focus* es una palabra o secuencia de palabras que definen una pregunta y la desambiguan indicando lo que está buscando [10]. La presentación de los resultados de FREyA incluye un grafo para su visualización.

2.2 Gingseng

Ginseng (Guided Input Natural Language Search Engine) es otra ILN que se basa en una gramática de preguntas que se extiende dinámicamente por la estructura de una ontología para guiar a los usuarios en la formulación de consultas en un idioma aparentemente similar al inglés. Basándose en la gramática, Ginseng traduce las consultas al lenguaje SPARQL, que permite su ejecución [6].

Proporciona un acceso de consulta a cualquier base de conocimiento OWL. Guía al usuario a formular consultas por lo que no es necesario interpretarlas (lógica o sintácticamente) y no utiliza ningún vocabulario predefinido. Gingseng sólo conoce el vocabulario definido por las ontologías consideradas actualmente y el usuario tiene que seguirlo [6]. Esto puede limitar las posibilidades del usuario en general, pero asegura que todas las consultas puedan ser respondidas [6].

La arquitectura de Gingseng consta de tres partes: una gramática multinivel, un parser incremental y una capa de acceso a la ontología. La gramática multinivel consta de una parte estática que especifica las estructuras de oraciones de consulta generalmente posibles y una parte dinámica que se genera a partir de las ontologías utilizadas. Las reglas gramaticales estáticas proporcionan las estructuras y frases básicas para las preguntas en inglés. Las reglas gramaticales dinámicas se generan a partir de cada ontología cargada en Gingseng y son utilizadas para extender la parte de las gramáticas estáticas [6].

La gramática completa es utilizada por el parser incremental primero para proporcionar alternativas al usuario durante el ingreso de consultas, y segundo para almacenar información sobre como construir consultas SPARQL. Finalmente, el framework Jena es utilizado para la capa de acceso a las ontologías. Cuando se completa la ejecución de una consulta, Ginseng muestra la consulta SPARQL generada y los resultados para el usuario [6].

Ginseng, a diferencia de FREyA, su principal característica es que las consultas que realiza el usuario son controladas mediante una gramática basada en los conocimientos almacenados en la ontología. Esto da lugar a una de sus ventajas e igualmente a su desventaja, ya que la gramática le da el control sobre lo que escribe el usuario y así evitar consultas no válidas, pero a la vez limita las consultas que el usuario puede realizar.

2.3 QuestIO System

QuestIO (Question-Based Interface to Ontologies) es una ILN para acceder a información estructurada, es independiente del dominio y es fácil de usar sin formación [7]. Es de dominio abierto (o personalizable a nuevos dominios con muy poco costo), con el vocabulario no predefinido, es decir, que se deriva automáticamente de los datos existentes en la base de conocimientos [7]. El sistema trabaja convirtiendo consultas en lenguaje natural a consultas formales de tipo SeRQL [11] (aunque se pueden usar otros lenguajes de consulta).

El proceso de funcionamiento de QuestIO empieza por procesar la ontología del dominio, creando de manera automática un diccionario léxico a partir del conocimiento obtenido. A partir de la ontología el diccionario es capaz de identificar menciones de clases, propiedades, instancias y valores de propiedad asociados con las instancias [11].

Cuando QuestIO recibe una consulta, el sistema realiza los siguientes pasos:

- Realiza un análisis lingüístico que consiste en un análisis morfológico del texto por medio de un tokenizador y herramienta de etiquetado.
- La segunda fase es ejecutar el diccionario ontológico creado al iniciar el sistema sobre el texto de la consulta. Esto crea anotaciones para todas las menciones que el diccionario pudo identificar clases, propiedades, instancias y valores de propiedades de tipos de datos.
- Inicia un proceso de transformación iterativo para convertir el texto de entrada en una consulta formal. Primero separa el texto de entrada en tokens, determinando la parte del discurso de cada uno y agregando anotaciones con su raíz morfológica. Después trata de identificar en el texto de entrada las menciones de los recursos de la ontología para poder generar una consulta formal en SeRQL.
- Finalmente ejecutar la consulta en la base de conocimiento y desplegar los resultados.

QuestIO es una ILN que a diferencia de FREyA y Gingseng, utiliza el lenguaje de consultas SeRQL pudiendo ser una desventaja ya que el estándar oficial para consulta de ontologías en la Web Semántica es el lenguaje SPARQL. Otra desventaja es que depende de una ontología bien diseñada para su funcionamiento, lo cual generalmente requiere de un especialista del dominio para el diseño de la ontología, lo que lo hace menos accesible para usuarios no especializados.

2.4 ORAKEL

ORAKEL es una ILN a base de conocimientos que convierte las preguntas a forma lógica [8] ya que para la representación del conocimiento utiliza el lenguaje f-logic [12]. Recibe como entrada una consulta de lenguaje natural que es convertida a una fórmula lógica de primer orden. Después la fórmula lógica es transformada al lenguaje de consulta específico, que puede ser SPARQL o el lenguaje de consultas de f-logic [13]. ORAKEL tiene los siguientes componentes principales:

1. Léxico del dominio y léxico general, ambos creados por el experto del dominio.
2. Base de conocimiento, compuesta por la ontología del dominio.
3. FrammeMapper, es un experto en el dominio que debe conocer la base de conocimientos subyacente.
4. Intérprete de consultas (*Query interpreter*), construye una consulta formulada por el usuario a forma lógica con respecto a los predicados del dominio.
5. Convertidor de la consulta (*Query converter*). Está implementado en lenguaje Prolog. Este componente recibe consultas en forma lógica y las traduce al lenguaje de la base de conocimientos (f-logic).
6. Generación de respuesta (*Answer generation*).

A diferencia de las otras ILN expuestas en este trabajo, ORAKEL es compatible también con el lenguaje f-logic para la representación del conocimiento. Su principal ventaja es que fue diseñado para portar ILN's entre dominios de manera eficiente. La desventaja es que requiere del conocimiento de un experto del dominio para la generación del léxico.

2.5 AquaLog

Es un sistema de pregunta-respuesta portable que toma como entrada consultas expresadas en lenguaje natural y una ontología y devuelve respuestas extraídas del marcado semántico compatible con la ontología disponible [9]. La arquitectura de AquaLog se puede caracterizar como un modelo en cascada en el que una consulta en lenguaje natural se traduce a un conjunto de representaciones intermedias basadas en tripletas. Dichas tripletas son traducidas a tripletas compatibles con la ontología. El modelo tripletas que usa AquaLog es de la forma (sujeto, predicado, objeto) [9]. Los módulos principales de AquaLog son el componente lingüístico y el servicio de similitud de relación.

La función del componente lingüístico es convertir la consulta de lenguaje natural a una consulta en tripletas. AquaLog utiliza la infraestructura y los recursos de GATE [14] para analizar la pregunta como parte del componente lingüístico. GATE devuelve un conjunto de anotaciones sintácticas asociadas con la consulta de entrada. Estas anotaciones incluyen información sobre oraciones, tokens, sustantivos y verbos. AquaLog extiende el conjunto de anotaciones devueltas por GATE, identificando términos, relaciones, indicadores de preguntas (qué/quién/cuándo/etc.) y patrones o tipos de preguntas.

AquaLog presenta una solución en la que se combinan diferentes estrategias para dar sentido a una consulta en lenguaje natural con respecto al universo del discurso cubierto por la ontología. Utilizando el framework GATE le da capacidad para mejorar el procesamiento de las consultas en lenguaje natural.

Las interfaces de lenguaje natural analizadas anteriormente utilizan ontologías para almacenar conocimiento, el cual se almacena en forma de archivos xml y se extrae mediante un lenguaje de consulta. A excepción de QuestIO (que utiliza SeRQL), las demás interfaces utilizan el lenguaje SPARQL que es el estándar oficial para usarse en la Web Semántica.

3. Arquitectura propuesta

La interfaz que actualmente se tiene en desarrollo, recibe una consulta formulada en lenguaje natural por los usuarios en idioma inglés, la traduce al lenguaje de consultas de ontologías SPARQL [3] y mediante ésta, extrae de la ontología del usuario la información solicitada. Su arquitectura está dividida en dos módulos principales: a) el módulo generador de conocimiento (MGC) y b) el módulo de la interfaz (MI). El MGC se encarga de generar el conocimiento que el MI necesita para responder a las consultas formuladas en lenguaje natural por los usuarios. En la sección 3.1 se describe el MGC y en la sección 3.2 se describe el MI. En la Fig. 1 se presenta la arquitectura general de la interfaz propuesta.

3.1 Módulo generador de conocimiento

Para que el MI sea capaz de responder consultas formuladas en lenguaje natural, es necesario proveerla de conocimiento acerca de la estructura de la ontología que el usuario desea consultar y acerca del dominio lingüístico conversacional. Con el fin de facilitar su portabilidad a diferentes ontologías del usuario, el MGC es el encargado de generar este conocimiento, buscando minimizar la intervención del usuario al configurar el MI.

Como primer paso, el usuario indica al MGC la ontología que desea consultar. Posteriormente, el MGC accede a la ontología y analiza su estructura, es decir, identifica las clases, propiedades de objeto, propiedades de datos y la manera en que estos elementos se encuentran relacionados. Para este fin, se utilizó el lenguaje de consulta SPARQL, en el lenguaje de ontologías OWL [2]. A través de consultas formuladas en SPARQL se pueden identificar los elementos que componen la estructura de una ontología y se puede acceder a la información almacenada, generalmente modelada como individuos, es decir, como instancias de las clases en la ontología.

Como siguiente paso, los elementos identificados son modelados semánticamente utilizando una representación semántica definida a priori para este propósito. La representación semántica se diseñó primeramente en el software Protégé [15]. Posteriormente, para integrar la estructura de esta representación semántica en el MGC, se utilizó el framework Apache Jena [16], el cual permite gestionar ontologías desde el lenguaje de programación Java, en el cual está diseñada la interfaz propuesta.

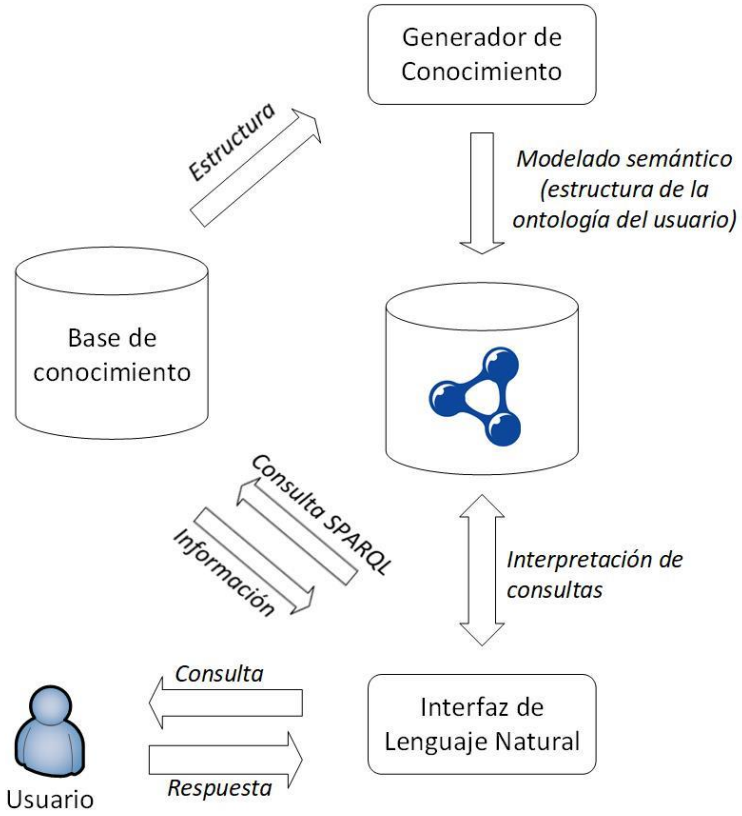

Fig. 1. Arquitectura general de la interfaz propuesta.

A continuación, cada uno de los nombres de los elementos modelados son adicionados con sinónimos, así como con palabras que compartan el mismo lema. Esto se realiza con el fin de proveer al MI de conocimiento lingüístico acerca del dominio de la ontología del usuario. El vocabulario generado también es agregado al modelado semántico.

Por último, después de generar el modelado semántico, el MGC lo almacena en una ontología, la cual es creada dinámicamente. Esta ontología generada por el MGC, contiene el conocimiento que el MI requiere para responder a consultas formuladas en lenguaje natural por los usuarios. En la Fig. 2 se presenta la arquitectura correspondiente al MGC y en la Fig. 3 se muestra un fragmento del modelado semántico generado.

El fragmento presentado fue generado a partir de la ontología Geography.owl. Utilizada en la evaluación de la interfaz FREyA por Damljanovic [5]. A su vez, esta ontología fue generada a partir de la base de datos deductiva de Raymond J. Mooney [17], la cual contiene información acerca de la geografía de Estados Unidos y fue distribuida como datos de ejemplo en Turbo Prolog 2.0. En la Fig. 4 se muestra la estructura de la ontología Geography.owl.

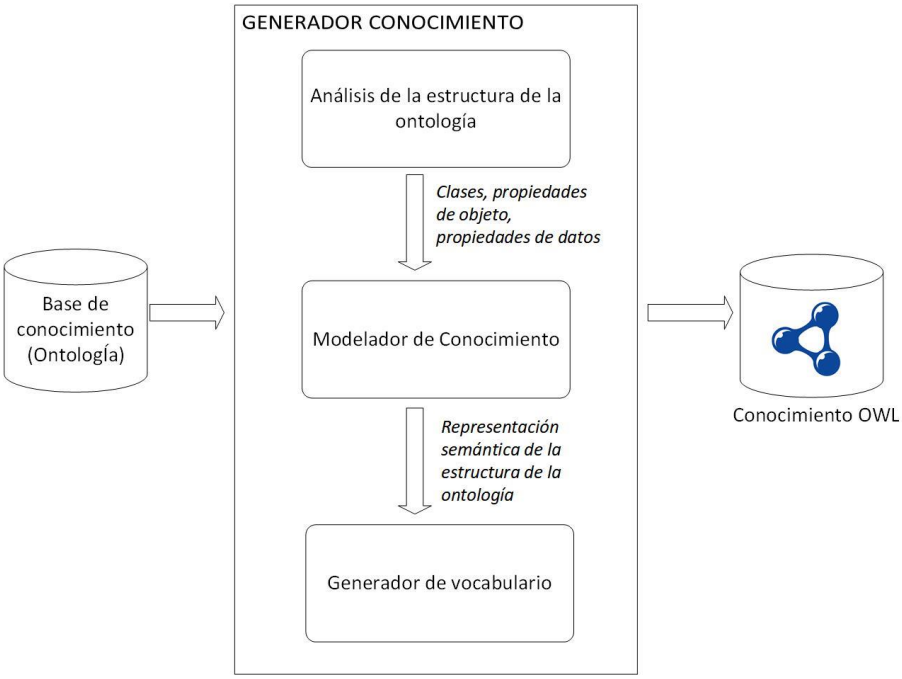

Fig. 2. Arquitectura del módulo generador de conocimiento.

La representación semántica de la Fig. [3] está conformado por las clases *CClass*, *CObjectProperty*, *CDatatypeProperty* y *CToken*. Además de las propiedades de objeto *hasDatatypeProperty*, *mappingObject*, entre otras.

Las clases *State*, *Lake*, *City*, *Capital*, *Lake*, *LoPoint*, *Mountain*, *HiPoint*, *River* y *Road*, mostradas en la Fig. [4], son modeladas en la Fig. [3] como individuos de la clase *CClass*.

Las propiedades de objeto *Borders*, *isCityOf*, *hasCity*, *isCapitalOf*, *hasCapital*, *isLakeOf*, *hasLake*, *isLowestPointOf*, *hasLowestPoint*, *isMountainOf*, *hasMountain*, *isHighestPointOf*, *hasHighestPoint*, *runsThrough*, *hasRiver*, *passesThrough* y *hasRoad*, mostradas en la Fig. [4], son modeladas en la Fig. [3] como individuos de la clase *CObjectPropery*.

Las propiedades de datos *statePopulation*, *stateArea*, *abbreviation*, *statePopDensity*, *cityPopulation*, *lakeArea*, *loElevation*, *height*, *hiElevation*, *length* y *number*, mostradas en la Fig. [4], son modeladas en la Fig. [3] como individuos de la clase *CDatatypeProperty*.

La clase *CToken* es utilizada para modelar el vocabulario de la interfaz. Considerando la Fig. [3], si el usuario introduce la palabra "*population*", el MI sabrá que posiblemente se está refiriendo a los individuos *STATE* (CClass), *STATEPOPULATION* (CObjectProperty) o *STATEPOPDENSITY* (CObjectProperty). De esta manera el MI utiliza el conocimiento generado para interpretar las consultas de los usuarios y generar la consulta SPARQL.

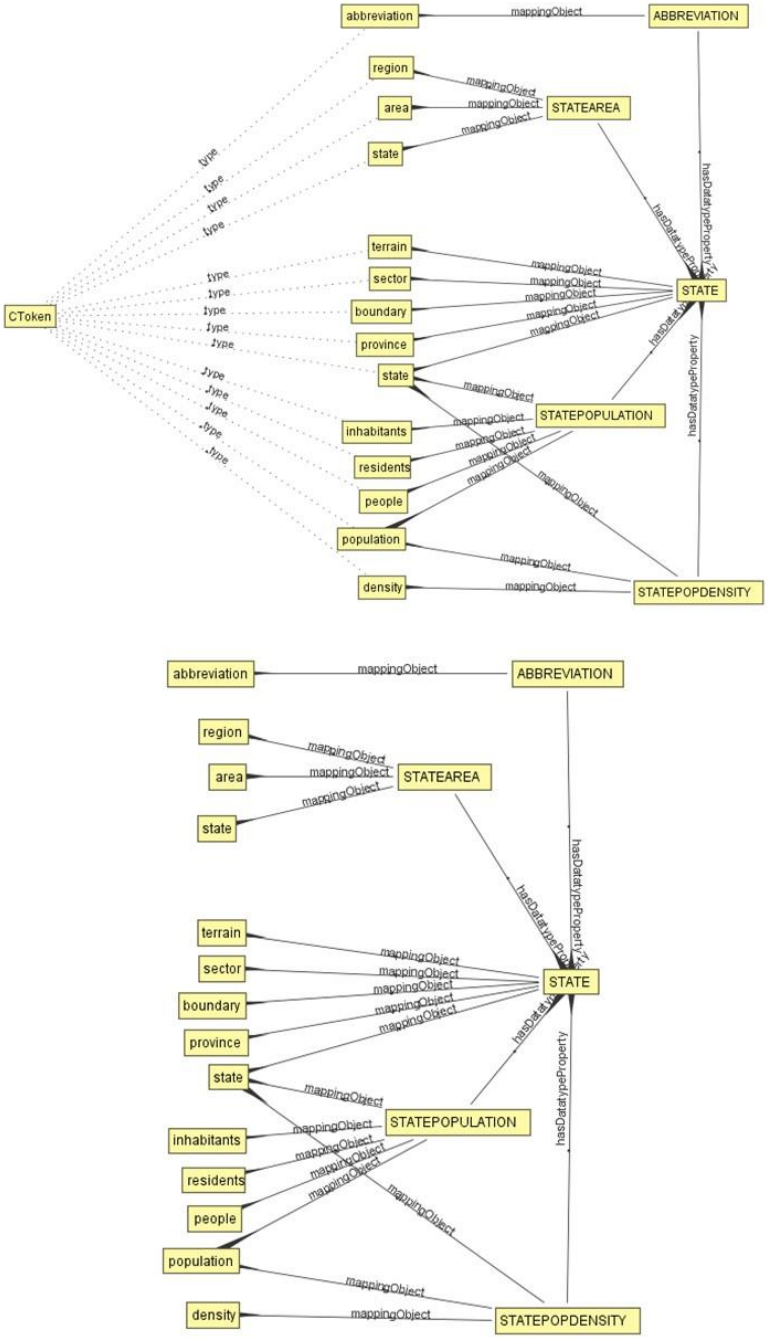

Fig. 3. Fragmento del modelado semántico generado.

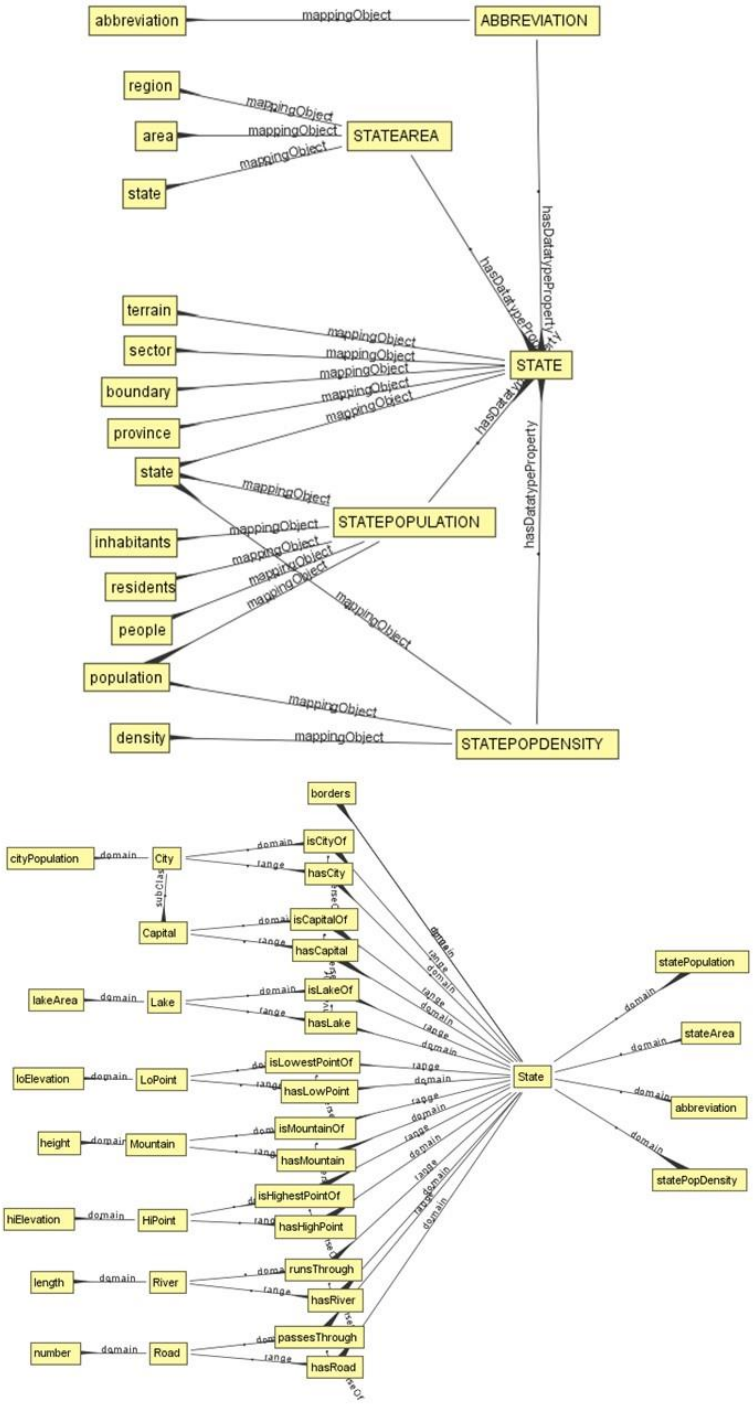

Fig. 4. Estructura de la ontología Geography.owl.

3.2 Módulo de la interfaz

El MI tiene como objetivo traducir la consulta formulada en lenguaje natural por el usuario a una consulta SPARQL, con la cual extraerá de la ontología del usuario la información solicitada. Su funcionamiento es mostrado en la Fig. [5].

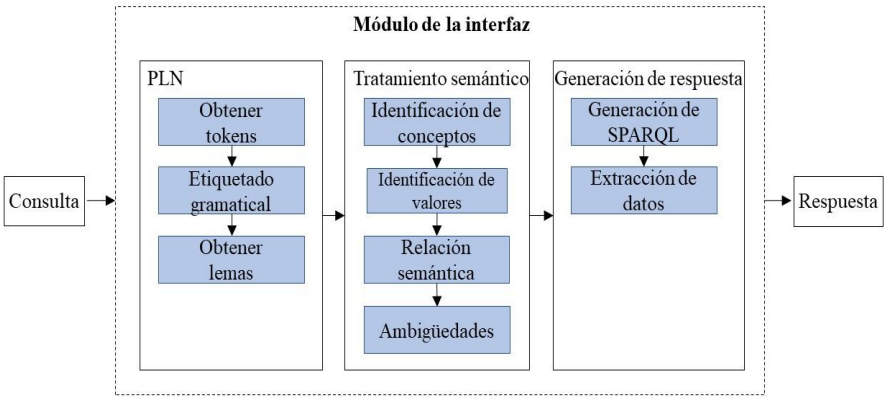

Fig. 5. Funcionamiento del módulo de la interfaz.

El primero paso efectuado por el MI para generar la consulta SPARQL, es realizar a la consulta introducida por el usuario, una fase de procesamiento de lenguaje natural. Esta fase consiste en separar la consulta del usuario en un conjunto de palabras o *tokens* aislados. Posteriormente, estos tokens son etiquetados de acuerdo a su función gramatical dentro de la consulta del usuario. Por último, se obtiene el lema de cada uno de los tokens. Esta fase es realizada utilizando Freeling [18], la cual es una suite de herramientas para el procesamiento de lenguaje natural que ofrece soporte para diversos idiomas, entre ellos, inglés y español. Suponiendo que el usuario introduce la consulta "*What is the population density of Wyoming?*", el resultado de esta fase es como se muestra en la Tabla 1.

Tabla 1. Ejemplo del resultado del procesamiento de lenguaje natural.

Token	Lema	PoS	Significado PoS
What	what	WP	pos= pronoun type= interrogative
is	be	VBZ	pos= verb vform =personal person=3
the	the	DT	pos= determiner
population	population	NN	pos=noun type=common num=singular

density	density	NN	pos=noun type=common num=singular
of	of	IN	pos=preposition
Wyoming	wyoming	NP	pos=noun type=proper
?	?	Fit	pos=punctuation type=questionmark punctenclose=close

El segundo paso que realiza el MI es un tratamiento semántico. En esta fase, se analiza cada token que haya sido etiquetado gramaticalmente como un sustantivo (noun) o un adjetivo (adjective). El análisis consiste en identificar si algún token *mapea* a algún concepto, es decir, a algún individuo de la clase *CToken* de la ontología generada por el MGC (tal como se mostró en el ejemplo de la palabra *population* al final de la sección 3.1), lo cual se realiza a través de una coincidencia exacta de caracteres. Mediante el modelado presentado en la Fig. [3], el MI identifica si el mapeo hace referencia a una clase, propiedad de objeto o propiedad de datos. Si alguno de estos tokens no mapea a algún elemento en la ontología, se realiza una búsqueda en la ontología del usuario para identificar si se trata de algún valor. En la consulta del ejemplo anterior, *Wyoming* se considera como un valor, ya que el MI identifica que es un individuo instanciado a partir de la clase *State*. Es importante mencionar que identificar correctamente los mapeos y los valores, es esencial para generar la consulta SPARQL. En la Tabla 2 se presenta el resultado de este paso.

Tabla 2. Ejemplo de identificación de mapeos y valores.

Token	Lema	PoS	Significado PoS
What	what	WP	
is	be	VBZ	
the	the	DT	
population	population	NN	Token=http://www.uacj_nlp.com/schema/nlp.OWL#population Class=http://www.uacj_nlp.com/schema/nlp.OWL#CDatatypeProperty Object=http://www.uacj_nlp.com/schema/nlp.OWL#STATEPOPULATION Token=http://www.uacj_nlp.com/schema/nlp.OWL#population Class=http://www.uacj_nlp.com/schema/nlp.OWL#CDatatypeProperty Object=http://www.uacj_nlp.com/schema/nlp.OWL#**STATEPOPDENSITY**

density	density	NN	Token=http://www.uacj_nlp.com/schema/nlp.OWL#citypopulation Class=http://www.uacj_nlp.com/schema/nlp.OWL#CDatatypeProperty Object=http://www.uacj_nlp.com/schema/nlp.OWL#CITYPOPULATION Token=http://www.uacj_nlp.com/schema/nlp.OWL#density Class=http://www.uacj_nlp.com/schema/nlp.OWL#CDatatypeProperty Object=http://www.uacj_nlp.com/schema/nlp.OWL#**STATEPOPDENSITY** Token=http://www.uacj_nlp.com/schema/nlp.OWL#population Class=http://www.uacj_nlp.com/schema/nlp.OWL#CDatatypeProperty Object=http://www.uacj_nlp.com/schema/nlp.OWL#**STATEPOPDENSITY**
of	of	IN	
Wyoming	wyoming	NP	Individual=http://www.mooney.net/geo#wyoming Class=http://www.mooney.net/geo#**State** Label=wyoming@en Individual=http://www.mooney.net/geo#wyomingMi Class=http://www.mooney.net/geo#**City** Label=wyoming@en
?	?	Fit	

Posteriormente, se intenta identificar si existe alguna relación semántica entre algunos de los tokens. Esto se determina si entre sus mapeos existe alguno en común, además de seguir algunas reglas establecidas a priori. En caso de ser afirmativo, ambos tokens son combinados en un token compuesto, conservando solamente los mapeos coincidentes o relacionados y desechando el resto de sus mapeos. Esto se realiza con el fin de simplificar el proceso de generación de la consulta SPARQL. Como se pudo ver en la Tabla 2, los tokens *population* y *density* tienen en común el mapeo a STATEPOPDENSITY, por lo cual, el MI los combina en un solo token.

Posteriormente, el MI procesa los valores identificados. Si un valor pertenece a más de una clase podría causar un problema de ambigüedad. En este caso, *Wyoming* pertenece a dos clases, *State* (*wyoming*) y *City* (*wyomingMi*), por lo que se intenta resolver la ambigüedad identificando la(s) clase(s) que concuerde(n) con las clases de los demás elementos de la consulta. Como el MI identificó en base a la Fig. 3 que STATEPOPDENSITY es una propiedad de datos que pertenece a STATE, se descarta la clase *City*. En caso de que el MI no pueda resolver la ambigüedad, muestra un diálogo

de clarificación al usuario para que éste la resuelva. En la Tabla 3 se presenta el resultado de esta fase de tratamiento semántico.

Tabla 3. Ejemplo del resultado de la fase de tratamiento semántico.

Token	Lema	PoS	Significado PoS
What	what	WP	
is	be	VBZ	
the	the	DT	
population density	population density	NN	Token=http://www.uacj_nlp.com/schema/nlp.OWL#population Class=http://www.uacj_nlp.com/schema/nlp.OWL#CDatatypeProperty Object=http://www.uacj_nlp.com/schema/nlp.OWL#**STATEPOPDENSITY**
of	of	IN	
Wyoming	wyoming	NP	Individual=http://www.mooney.net/geo#wyoming Class=http://www.mooney.net/geo#**State** Label=wyoming@en
?	?	Fit	

El tercer paso es generar la consulta SPARQL con los elementos identificados después de la fase de tratamiento semántico, mostrados en la Tabla 3 y utilizar esta consulta para extraer de la ontología del usuario la información solicitada. Para generar la consulta SPARQL se sigue una serie de reglas definidas a priori. Por ejemplo, en base a la Fig. 3 y a la Tabla 3, el MI identificó que STATEPOPDENSITY es una propiedad de datos relacionada con la clase STATE. Estos elementos son identificados en la ontología del usuario por el MI como <http://www.mooney.net/geo#statePopDensity> y como <http://www.mooney.net/geo#State>. Además, *Wyoming* es un individuo de la clase *State*. Por tal motivo, el MI genera la siguiente consulta SPARQL, donde la variable que forma parte de la cláusula *Select* se forma a partir del token combinado *population density*, el cual es transformado ?*population_density*. El resultado mostrado por el MI utilizando la consulta SPARQL generada se muestra en la Fig. 6.

```
Select ?population_density
Where {
       <http://www.mooney.net/geo#statePopDensity>
       <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
       <http://www.w3.org/2002/07/owl#DatatypeProperty> .

       <http://www.mooney.net/geo#wyoming>
       <http://www.mooney.net/geo#statePopDensity>
       ?population_density .

       <http://www.mooney.net/geo#wyoming>
       <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
       <http://www.mooney.net/geo#State>
}
```

population_density
0.20830059

Fig. 6. Resultado de la consulta "*What is the population density of Wyoming*" utilizando la ontología Geography.owl.

4. Implementación

En esta sección se describe la implementación de la ILN propuesta en este trabajo. Se describen los componentes de software y los dispositivos de hardware donde son alojados.

La ILN se basa en una arquitectura de tipo cliente-servidor, utiliza una aplicación Android como interfaz gráfica y un servidor de aplicaciones donde se realiza el procesamiento de las consultas. El servidor funciona mediante Java Server Pages (JSP) [19] como tecnología para su implementación por lo que utiliza Apache Tomcat [20] que es un contenedor de aplicaciones web basado en Java.

El diagrama de despliegue de la Fig. 7 muestra los componentes de software y los dispositivos de hardware utilizados. El sistema implementa los siguientes componentes de software: una aplicación Android, un controlador de eventos de interacción entre los componentes, una base de conocimientos, un componente para el procesamiento de lenguaje natural y otro para generar y ejecutar consultas SPARQL en la base de conocimiento.

El proceso general para cada consulta de entrada es de acuerdo con los siguientes pasos:

1. La aplicación Android recibe como entrada la consulta en forma de voz, la convierte a texto y la envía al controlador para su procesamiento.
2. El controlador recibe la consulta de texto y la envía al procesador LN.
3. El procesador LN recibe la consulta de texto y a partir de esta obtiene los tokens, etiquetas gramaticales y lemas. Finalmente envía al controlador un conjunto de tokens, etiquetas gramaticales y lemas.

4. Una vez que el controlador recibe el conjunto de tokens etiquetas y lemas, lo envía al módulo de conocimiento.
5. El módulo de conocimiento se encarga de generar consultas SPARQL y ejecutarlas en la ontología para obtener la respuesta. Por último, envía la respuesta al controlador.
6. El controlador envía la respuesta a la aplicación Android para mostrarla al usuario.

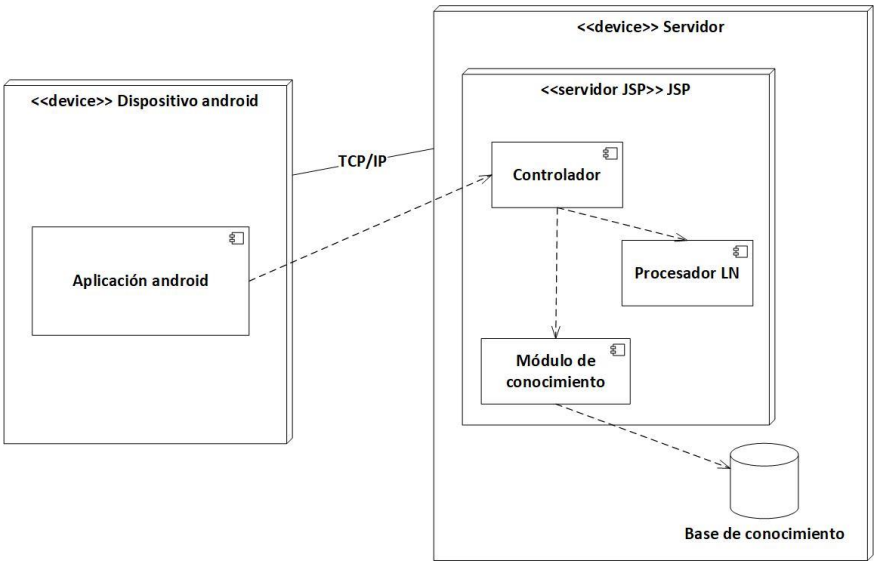

Fig. 5. Funcionamiento del módulo de la interfaz.

Para la implementación de cada componente se utilizaron varias herramientas de desarrollo. La aplicación Android fué implementada mediante el entorno de desarrollo integrado Android Studio [21], utilizando librerías nativas para convertir consultas de voz a texto. Se puede utilizar en dispositivos que utilicen Android como sistema operativo.

El controlador está implementado mediante la tecnología JSP [22]. Recibe y realiza peticiones a los otros componentes para gestionar todo el proceso del sistema, desde recibir la consulta hasta enviar la respuesta a la aplicación Android.

El procesador LN está implementado en lenguaje Java [23]. Después de revisar varias herramientas de procesamiento de lenguaje natural como la librería de código abierto Freeling [18], la librería Apache OpenNLP [24] y la herramienta Stanford CoreNLP [25], se eligió Freeling para implementar este componente ya que, a diferencia de las otras tiene mejor soporte para los idiomas inglés y español.

La base de conocimiento es almacenada en forma de ontología y en archivos de tipo OWL [2] y RDF [26], para acceder a ella se utiliza el lenguaje de consulta SPARQL. El módulo de conocimiento fue implementado utilizando el lenguaje de programación Java y Apache Jena [16] que es un framework Java para la consulta de bases de

conocimientos almacenadas en ontologías RDF y OWL utilizando SPARQL, y que permite gestionar la ontología en tiempo de ejecución.

La ILN permite seleccionar la ontología que se desea consultar previamente creada. Utilizando el framework Jena la ILN accede a la ontología y analiza su estructura para posteriormente responder las consultas en lenguaje natural de los usuarios.

5. Conclusiones

Con la aparición de la Web Semántica el uso de ontologías ha tomado demasiada importancia, ya que se proponen utilizar como medio de representación de conocimiento con el objetivo de facilitar la interoperabilidad de la información en la web. Las interfaces de lenguaje natural a ontologías son excelentes herramientas para acceder al conocimiento almacenado en ontologías. Su desarrollo podría facilitar a los usuarios la localización de recursos, la comunicación entre aplicaciones informáticas, la búsqueda de información, entre otras actividades realizadas en la Web.

En este trabajo se propone la arquitectura de una ILN que utiliza bases de conocimiento basadas en ontologías, la cual recibe consultas de voz desde un dispositivo Android, las convierte a texto, aplica técnicas de procesamiento de lenguaje natural y de representación de conocimiento para generar las consultas SPARQL correspondientes y extrae información de la ontología.

A pesar de que el modelado y el procesamiento semántico que realiza la ILN le han permitido responder a consultas formuladas en lenguaje natural por los usuarios, se han detectado áreas de oportunidad.

Como trabajos futuros a corto plazo, se propone mejorar el esquema de representación de conocimiento, así como el procesamiento semántico de la ILN. También se propone fortalecer el procesamiento de lenguaje natural para que la ILN sea capaz de responder consultas que integren comparaciones, negaciones, fechas y números. Posteriormente, se propone evaluar el desempeño de la ILN contra FREyA utilizando la ontología Geography.owl y sus respectivos corpus de consultas llamados Geoquery 250 y Geoquery 880.

Como trabajos futuros a largo plazo, se propone abordar problemas complejos relacionados con el lenguaje natural, tales como anáforas y posteriormente, elipsis intersentencial. La anáfora es cuando se hace referencia a una entidad mencionada anteriormente y su resolución es importante para responder a preguntas que se refieren a la misma entidad en diferentes formas. La elipsis ocurre cuando se omiten una o más palabras en una oración y que son sobrentendidas debido a que ya se han mencionado antes.

Agradecimientos

Agradecemos a PRODEP por el apoyo brindado al proyecto titulado "*Conocimiento Semántico en una Interfaz de Lenguaje Natural Portable para Acceder a Información de Bases de Datos Multidimensionales en el Área de Negocios Inteligentes*", con el cual, este artículo fue posible. UACJ-PTC-373.

Referencias

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34-43 (2001)
2. McGuinness, D.L., Van Harmelen, F.: OWL Web Ontology Language Overview. W3C recommendation 10, (2004)
3. Prud'hommeaux, E., Seaborne, A.: SPARQL 1.1 Overview. W3C recommendation 21, (2013)
4. Kaufmann, E., Bernstein, A.: How useful are natural language interfaces to the semantic web for casual end-users? En: Franconi, E., Kifer, M., May, W. (eds.). ESWC 2007, The Semantic Web, p. 281-294. Springer, Heidelberg (2007)
5. Damljanovic, D., Agatonovic, M., Cunningham, H.: Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction. En: Aroyo, L., Antoniou, G., Hyvonen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.). ESWC 2010, Extended Semantic Web Conference, vol. 6088, p. 106-120. Springer, Heidelberg (2010)
6. Bernstein, A., Kaufmann, E., Kaiser, C., Kiefer, C.: Ginseng: A guided input natural language search engine for querying ontologies. En: Jena User Conference, Citeseer (2006)
7. Tablan, V., Damljanovic, D., Bontcheva, K.: A Natural Language Query Interface to Structured Information. En: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. The Semantic Web: Research and Applications, p. 361-375. Springer Berlin Heidelberg (2008)
8. Cimiano, P.: Orakel: A natural language interface to an f-logic knowledge base. En: Meziane, F., Métais, E. (eds.), Natural Language Processing and Information Systems, p. 401-406. Springer Berlin Heidelberg, Heidelberg (2004)
9. Lopez, V., Pasin, M., Motta, E.: Aqualog: An ontology-portable question answering system for the semantic web. En: Gómez-Pérez, A., Euzenat, J. (eds.). The Semantic Web: Research and Applications. p. 546-562. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
10. Damljanovic, D., Agatonovic, M., Cunningham, H.: Identification of the Question Focus: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction. En: 7th Language Resources and Evaluation Conference (LREC). ELRA, La Valletta (May 2010)
11. Aduna, J.B., Aduna, A.K.: SeRQL: A Second Generation RDF Query Language. En: SWAD-Europe Workshop on Semantic Web Storage and Retrieval, p. 13-14. (2003)

12. Kifer, M., Lausen, G.: F-logic: A higher-order language for reasoning about objects, inheritance, and scheme. En: ACM SIGMOD Record, vol. 18, p. 134-146. ACM, New York, NY, USA (1989).
13. Cimiano, P., Haase, P., Heizmann, J.: Porting natural language interfaces between domains an experimental user study with the orakel system. En: Proceedings of the 12th international conference on Intelligent user interfaces, p. 180-189. ACM, New York, NY, USA (2008)
14. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. En: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), p. 168-175. Philadelphia (2002)
15. Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A.: WebProtege: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web. Semantic Web Journal, vol. 4, no. 1, p. 89-99. IOS Press (2013)
16. APACHE Jena, https://jena.apache.org/, _ultimo acceso 2017
17. Zelle, J. M., Mooney, R. J.: Learning to parse database queries using inductive logic programming. En: Proceedings of the 14th National Conference on Artificial Intelligence, p. 1050-1055. (1996)
18. Padró, L.: Analizadores Multilingües en Freeling. En: Lingüamatica, vol. 3, p. 13-20. (2011)
19. Hunt, J.: Java Server Pages. En: Java and Object Orientation: An Introduction, p3361-370. Springer London, London (2002)
20. Vukotic, A., Goodwill, J.: Introduction to Apache Tomcat 7. En: Apache Tomcat 7, p. 1-15. Apress, Berkeley, CA (2011)
21. Hohensee, B., Hidalgo, I.: Introducción a Android Studio. Incluye Proyectos Reales Y El Código Fuente. Babelcube Incorporated (2014)
22. Falkner, J., Jones, K.: Servlets and JavaServer Pages TM The J2EE TM Technology Web Tier. Addison-Wesley (2004)
23. Deitel, P., Deitel, H.: Java Cómo Programar. 9a edn. Pearson Education (2012)
24. Community, A.O.D.: Página de inicio Apache OpenNLP, https://opennlp.apache.org, último acceso 2018
25. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. En: Association for Computational Linguistics (ACL) System Demonstrations. p. 55-60. (2014)
26. Schreiber, G., Amsterdan, V.U., Raimond, I., BBC.: RDF 1.1 Primer. W3C Working Group Note 24, (2014)