2. MARCO TEÓRICO

Con este proyecto se pretende desarrollar un estado del arte sobre un punto muy importante dentro de lo que llamamos la Web 3.0. Las técnicas de recuperación de información basadas en ontologías aprovechan la Web semántica y permiten brindar una calidad de información más acertada a los usuarios que las búsquedas habituales.

2.1. INTRODUCCIÓN A LA RED SEMÁNTICA Y A LA ONTOLOGIA

La red semántica y la ontología son conceptos que relacionados dan un potencial bastante importante para la nueva tendencia de la Web. A continuación se introducirán estos conceptos con una breve definición.

2.1.1. Red semántica[16]

La red semántica es una extensión de la Web actual en la cual la información es estructurada y se brinda con un significado bien definido, que permite una comunicación más natural entre el usuario y los sistemas. La red semántica está basada en la idea de proporcionar a la Web datos definidos y enlazados de tal forma que puedan ser usados más efectivamente en el descubrimiento, automatización, integración y reutilización a través de diversas aplicaciones. Para que la Web alcance su máximo potencial debe evolucionar en la Web semántica proporcionando una plataforma de acceso universal que permita compartir y procesar los datos por herramientas automatizadas y por las personas.

[16] Hendler, James; Berners-Lee, Tim y Miller, Eric. “Integrating Applications on the Semantic Web”. Disponible en: http://www.w3.org/2002/07/swint.html. Consultado: Agosto 2009

2.1.2. Ontología[17]

Una ontología define los términos usados para describir y representar un área del conocimiento. Las ontologías son utilizadas por personas, bases de datos y aplicaciones que necesitan compartir un área del conocimiento ó simplemente un tema de un área específica como la medicina, gestión financiera, etc.

Las ontologías están compuestas principalmente por los siguientes conceptos:

- Clases
- Relaciones
- Propiedades (o atributos)

Las ontologías surgen de la necesidad de representar la semántica de los documentos y permitir que esta semántica sea utilizada por aplicaciones Web y agentes inteligentes. Algunas herramientas basadas en ontologías pueden realizar razonamientos automáticos, de este modo proveen servicios avanzados para aplicaciones inteligentes tales como agentes de software, bases de datos inteligentes y comercio electrónico. Usar ontologías implica que las aplicaciones del mañana puedan ser "inteligentes", en el sentido que pueden trabajar mucho más precisas bajo un nivel conceptual humano.

2.1.2.1 Metodologías para construir ontologías. Existen diferentes metodologías que ayudan a construir una ontología de mayor calidad. Algunos de los criterios para evaluar la calidad de una ontología son cobertura de un dominio en particular, complejidad y granularidad de la cobertura, casos de uso específicos, escenarios, requerimientos, aplicaciones y fuentes de datos para la cual fue desarrollada y propiedades formales tales como la consistencia y la

[17] Obrst, Leo; Ashpole, Benjamin; Ceusters, Werner; Mani, Inderjeet; Ray, Steve y Smith, Barry. "The Evaluation of Ontologies: Toward Improved Semantic Interoperability". Chapter in: Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences, Christopher J. O. Baker and Kei-Hoi Cheung, Eds., Springer, 2007.

completitud. Las ontologías también pueden ser evaluadas por preguntas como las siguientes: ¿Es la ontología mapeable a alguna ontología superior específica, de tal manera que su evaluación sea por lo menos parcialmente dependiente de esta última?, ¿Qué tipo de métodos de razonamiento pueden invocarse desde la ontología?[18]

La mayoría de las ontologías existentes han sido construidas manualmente. Construir ontologías de esta manera ha sido la aproximación que han tenido la mayoría de los ingenieros de ontologías. Sin embargo, este proceso consume mucho tiempo, propenso a errores y posee problemas en mantenimiento y actualización de las ontologías[19].

Es por esta situación que los investigadores han empezado a buscar y crear diferentes formas para construir ontologías de manera eficiente y sencilla. Así es como aparece la construcción semi automática y automática de las ontologías. A continuación veremos una de las metodologías utilizadas para tal fin.

InfoSleuth (MCC)[20]. Este proyecto de investigación desarrollado por MCC (Microelectronics and Computer Technology Corporation) busca desarrollar y desplegar nuevas tecnologías para encontrar información disponible en redes corporativas y externas. Se centra en los problemas de localización, evaluación, recuperación y fusión de la información en un entorno en el cual las nuevas fuentes de información están constantemente siendo agregadas. Este proyecto apunta a la construcción de una arquitectura de agente basada en ontologías.

El método utilizado para la generación automática de la ontología adoptada en este proyecto se describe a continuación:

---

[18] Ibid

[19] Ding, Ying y Foo, Schubert. "Ontology Research and Development (Part 1 - A Review of Ontology Generation)". Journal of Information Science, 2002.

[20] Ibid

Expertos humanos proporcionan al sistema un número pequeño de palabras semilla que representan los conceptos de alto nivel. Los documentos relevantes serán conseguidos desde la Web automáticamente.

El sistema procesa los documentos entrantes, extrae las frases que contienen las palabras semilla, genera los correspondientes términos concepto y los coloca en el lugar correcto en la ontología. Al mismo tiempo recoge candidatos para palabras semilla para la siguiente vuelta de procesamiento. Las iteraciones continúan hasta que algún resultado deseado sea alcanzado.

Varios tipos de relaciones son extraídos: "es-un", "parte-de", "fabricado-por", "propiedad-de", etc. La relación "asociado-con" es utilizada para definir las relaciones que no son del tipo "es-un".

Para cada iteración un experto humano es consultado para comprobar la exactitud de los conceptos. Si es necesario el experto tiene el derecho a realizar la corrección y reconstrucción de la ontología. A continuación se muestra un ejemplo de la estructura de una ontología generada:

Display
    Field emission display
        Diamond field emission display
    Flat panel display
    *display panel
        *display panel substrate
image
    video image
        *video image retrieval
            *multiaccess video retrieval server

Nota: El * muestra un concepto extra agregado al final del concepto de un nivel superior.

En el ejemplo anterior la identación muestra la jerarquía (relaciones entre clases y subclases). De este modo, 'field emission display', 'flat panel display' y 'display panel' son subclases de 'display'. Una regla obvia para generar esta jerarquía es que si la frase tiene 'display' como la última palabra en la frase, esta frase se convertirá en subclase de 'display'. Esta misma regla es aplicada en 'image'. Otra regla es que si la frase tiene 'display' como la primera palabra en la frase, esta frase también se convertirá en la subclase de 'display' con un '*' como indicador, tales como 'display panel', 'video image retrieval', etc.

Este sistema tiene varias características especiales:

- El esquema "descubre-y-notifica" le permite al sistema expandir la ontología con nuevos conceptos aprendiendo de los nuevos documentos y notificando a los humanos expertos de los cambios.
- El esquema "Atributo-relación-descubrimiento" permite descubrir algunos de los atributos asociados con ciertos conceptos. Por ejemplo, el método puede descubrir los atributos de las dimensiones físicas o el número de píxeles y siempre puede aprender el rango de sus posibles valores. Basado en los caracteres lingüísticos la relación "asociado-con" puede ser identificada automáticamente.
- Indexación de documentos: mientras se construye la ontología este método indexa los documentos para una recuperación futura. Opcionalmente guarda los resultados en una base de datos relacional. El sistema recoge "líneas de contexto" para cada concepto generado, mostrando como el concepto fue usado en el texto, así como la frecuencia y co-ocurrencia estadística para el descubrimiento de asociaciones entre palabras y minería de datos.

- El sistema le permite al usuario decidir entre la precisión y la completitud a través de la navegación de la ontología y las interfaces basadas en las relaciones "es-un" y "asociado-con".

Este sistema usa un etiquetado simple llamado parte-del-lenguaje (POS, part-of-speech) para dirigir el análisis sintáctico superficial (técnicas de extracción de información superficiales). La relación del concepto es detectada con base en las características lingüísticas. Como en cualquier otra aproximación basada en el cuerpo de los escritos, entre más rico y más completo sea el conjunto de datos, más alta será la confiabilidad de los resultados logrados como un resultado directo de la aplicabilidad de técnicas de aprendizaje automático.

2.1.2.2 Uso de las ontologías en la Web[21]. Las ontologías pueden ser usadas para mejorar las aplicaciones actuales basadas en Web y pueden permitir nuevos usos de la Web. A continuación se presentan algunos usos representativos de las ontologías en la Web.

Portales Web. Un portal Web es un sitio que provee contenido sobre un tema de interés. Un portal Web permite a los individuos interesados en el tema recibir noticias, encontrar y hablar con alguien, construir una comunidad y encontrar enlaces a recursos Web que estén relacionados con el tema del portal Web. Para que un portal Web sea exitoso debe de tener un punto de partida para localizar los contenidos. Normalmente este contenido es enviado por miembros de la comunidad ó se utiliza un medio de recolección de contenidos basado en etiquetas con información que pueda facilitar la búsqueda de la misma.

Sin embargo una simple indexación del contenido de áreas temáticas no le permite a la comunidad hacer búsquedas lo suficientemente eficientes como

[21] "OWL Web Ontology Language Use Cases and Requirements". Disponible en: http://www.w3.org/TR/webont-req/ Consultado: Agosto 2009.

realmente se requieren. Para permitir más inteligencia en la búsqueda de contenidos, los portales Web pueden definir una ontología para su comunidad. Esta ontología suministra una terminología para describir el contenido y axiomas que definen términos utilizando otros términos desde la ontología, es así como podríamos incluir terminologías como “Revista”, “Publicación”, “Persona” y “Autor”, haciendo que dicha ontología pudiera incluir definiciones como “Todas las revistas son publicaciones” ó “Los autores de las publicaciones son personas”.

Cuando se combinan estos hechos con las definiciones se permiten que otros hechos sean necesariamente verdaderos para ser inferidos. Estas inferencias permiten al usuario obtener resultados de búsquedas en el portal que son difíciles de obtener desde los sistemas convencionales para la recuperación de información.

Colecciones de multimedia. La ontología puede ser utilizada para proveer anotaciones semánticas para colecciones de imágenes, audio y otros objetos no textuales. La dificultad con los recursos multimedia es que normalmente son indexados a través de etiquetas, pero diferentes personas pueden describir estos objetos no textuales de una manera muy diferente. Idealmente las ontologías capturarían conocimiento adicional acerca del dominio que se pueden utilizar para mejorar la recuperación de recursos multimedia.

Gestión de sitios Web corporativos. Las grandes corporaciones normalmente tienen numerosas páginas relacionadas a comunicados de prensa, oferta de productos, procedimientos corporativos, reuniones informativas, comparaciones de productos internos y descripción de procesos. Las ontologías pueden ser utilizadas para indexar estos documentos y proveer un mejor medio de recuperación de la información. Una forma que incrementa la riqueza de las descripciones brindadas por las ontologías es trabajar con múltiples ontologías de manera simultánea, ya que cuando se utiliza una única ontología se limita muchas veces a un solo punto de vista ó a una pequeña parte de un dominio específico.

Documentación de diseño. Actualmente hay una gran cantidad de información que proviene de la documentación hecha para diseños de ingeniería como la industria aeroespacial, desarrollo de aplicaciones, entre otras. Este conjunto de documentos son construidos generalmente bajo una estructura de jerarquía. Las ontologías pueden construir un modelo de información que permita la exploración en términos de lo ítems que fueron representados, las asociaciones entre los ítems, las propiedades de los ítems y las relaciones a la documentación que los describe y define.

Agentes y servicios. La Web semántica provee agentes con la capacidad de entender e integrar diversos recursos de información. Un ejemplo concreto es el de un planeador de actividades sociales, el cual puede tomar las preferencias del usuario y usar esta información para planear las actividades que el usuario realizará en la noche. La tarea de planear estas actividades dependerá de la riqueza del entorno de servicio que está siendo ofrecido y de las necesidades del usuario. Este tipo de agentes requieren un dominio de ontologías para representar los términos como restaurantes, hoteles, etc. y servicios ontológicos que representen los términos usados en los servicios actuales.

Las ontologías habilitarán la captura de información necesaria para aplicaciones con el fin de discriminar y balancear entre las preferencias del usuario. Dicha información puede ser suministrada por un número de fuentes tales como portales, sitios de servicios específicos y la Web en general.

2.2. LENGUAJES DE BÚSQUEDA

Los lenguajes de búsqueda brindan la posibilidad de convertir lo que el usuario quiere buscar en una forma que sea entendida por un agente ó un motor de búsqueda. En algunas ocasiones hará las transformaciones necesarias para realizar dicha búsqueda dentro de las ontologías definidas.

En este punto tenemos dos aproximaciones. La primera es la utilización de la sintaxis del lenguaje con el que se está haciendo la búsqueda (el lenguaje de búsqueda sería el lenguaje natural) y hay otra aproximación que son lenguajes artificiales con una sintaxis definida que debe cumplir el usuario con el fin de llegar a las respuestas que requiere.

Para ser más específicos tenemos el ejemplo de cuando deseamos buscar en Google la definición de un término. Por ejemplo, casa. En un lenguaje natural podríamos buscar por la palabra "casa" simplemente, pero dado un lenguaje de búsqueda con una sintaxis definida la mejor aproximación para la búsqueda es define:casa.

A continuación mostraremos dos sistemas que utilizan lenguajes de búsqueda propios para que el usuario le indique al motor de una forma más específica lo que desea buscar, esta técnica es mucho más precisa en el contenido presentado al usuario con respecto al sistema tradicional de las palabras claves.

2.2.1. Lenguaje de búsqueda en WI OntoSearch[22].

Existen maneras más óptimas de entrar datos al motor de acuerdo a las ontologías que se requieran. El sistema ofrece los siguientes formatos para la entrada de mensajes.

Palabra única: para adquirir más ontologías candidatas una herramienta léxica como WordNet puede ser usada en el sistema.

Múltiples palabras: Se pueden ordenar múltiples palabras en un conjunto de conceptos $(I_1, I_2, \ldots, I_m)$.

[22] Gao, Mingxia; Liu, Chunnian y Chen, Furong. “An Ontology Search Engine Based on Semantic Analysis”. Proceedings of the Third International Conference on Information Technology and Applications - Volume 02, 2005, p. 256 - 259.

Segmento de ontología: Para crear conjuntos de conceptos, el segmento de ontología puede ser parseado con un parser especial.

2.2.2. Lenguaje de búsqueda en Swoogle[23]

Este motor de búsqueda de documentos semánticos utiliza un lenguaje creado para los usuarios que desean realizar búsquedas avanzadas en las cuales especifican algunas características o propiedades especiales de la información que requieren.

A continuación se presenta la sintaxis de las propiedades que el usuario puede especificar en su búsqueda avanzada:

- Un solo término: una sola palabra como "persona" o "comida". Por ejemplo, $def:persona$.
- Frase: un grupo de palabras con comillas doble. Por ejemplo, $desc:"helloworld"$. Tener en cuenta que el orden de las palabras en la frase son respetados.
- Búsqueda por intervalos: permiten al usuario encontrar documentos donde su propiedad contiene valores definidos por una cota inferior y superior. Por ejemplo, $hasDateLastmodificd:[2007-07-08.TO.2007-07-10]$.
- Búsquedas con caracteres comodín: para realizar la búsqueda con un solo caracter comodín se utiliza el símbolo $?$. Para realizar búsquedas con múltiples caracteres comodín se usa el símbolo $*$. Por ejemplo, $hasEncoding:SHIFT*$.
- Operadores booleanos: permite combinar términos a través de operadores lógicos. Los operadores soportados son AND, OR y NOT. Tener en

[23] "How to Search Semantic Web Documents/Ontologies". Disponible en: http://swoogle.umbc.edu/index.php?option=com_swoogle_manual&manual=search_swd. Consultado: Abril 2010.

cuenta que si se buscan dos términos y no se especifica operador lógico se tomará por defecto el operador lógico AND.

A continuación se muestran ejemplos utilizando las propiedades anteriores para realizar búsquedas avanzadas por los usuarios:

- $url{:}foaf$: busca documentos que contienen la palabra "foaf" como parte de sus URLs.
- $url{:}"http://www.w3.org/2000/01/rdf-schema"$: busca un SWD particular con la URL "http://www.w3.org/2000/01/rdf-schema".
- $desc{:}timbl$: busca documentos que tengan "timbl" en sus anotaciones de documento.
- $def{:}food$: busca documentos implícitamente que definen el término (clases/propiedades) que incluye "food".
- $food NOT web$: busca documentos relevantes que contienen "food" pero no contienen "web".

2.3. LENGUAJES UTILIZADOS EN ONTOLOGÍAS

En la Web Semántica existen varios formalismos utilizados para la representación del conocimiento. Se pueden enumerar los siguientes:

- Frames (Clases)
- Formalismos Lógicos de primer orden
- Descripciones lógicas (Axiomas)
- Grafos conceptuales
- Redes Semánticas

Para cada uno de los anteriores formalismos se han desarrollado lenguajes especializados entre los que se encuentran:

- KIF
- OKBC
- OCML
- LOOM
- FLogic
- SHOE
- XOL
- OIL
- DAML + OIL
- OML/CKML
- RDF(S)
- OWL

En la tabla 1 se puede ver cada lenguaje en que está especializado y que formalismo para la representación del conocimiento maneja.

Tabla 1. Formalismos en los lenguajes ontológicos

Formalismo / Lenguaje	Frames (Clases)	Lógicos de Primer Orden	Descripciones Lógicas (Axiomas)	Grafos Conceptuales	Redes Semánticas
KIF	■	■			
OKBC	■				
OCML	■	■			
LOOM			■		
FLogic	■	■			
SHOE	■				
XOL	■				
OIL	■		■		
DAML + OIL	■		■		
OML/CKML				■	
RDF(S)	■				■
OWL			■		

En los últimos años gran cantidad de lenguajes ontológicos han sido adoptados. W3C recomienda OWL y RDF para construir ontologías. Estas especificaciones de lenguajes fueron desarrolladas por muchos años dentro y fuera de la organización. OWL ha reemplazando sus predecesores y se ha convertido en el favorito a la hora de construir ontologías. A continuación se mostrará brevemente cada uno de los antecesores de OWL:

2.3.1 KIF (Knowledge Interchange Format)[24]

Lenguaje para el intercambio de conocimiento entre diferentes sistemas computacionales. Fue originalmente creado por Michael Genesereth y otros participantes en el DARPA Knowledge Sharing Effort. En el diseño de KIF se tuvieron en cuenta principalmente las siguientes tres características:

1. Tiene semánticas declarativas. Es posible entender el significado de las expresiones en el lenguaje sin apelar a un interpretador para manipular esas

[24] "Knowledge Interchange Format (KIF)". Disponible en: http://www.ksl.stanford.edu/knowledge-sharing/kif/ Consultado: Agosto 2009.

expresiones. De esta forma KIF difiere de otros lenguajes como Emysin y Prolog basados en interpretadores específicos.

2. Es lógicamente comprensible. Provee formalismos lógicos de primer orden, provee la definición de objetos, funciones y relaciones. Además de permitir el uso de expresiones de sentencias lógicas arbitrariamente. De este modo, difiere de lenguajes relacionales como SQL o lenguajes lógicos como Prolog.

3. Provee la representación de conocimiento sobre conocimiento. Permite al usuario representar el conocimiento explícitamente y permite introducir nuevas representaciones de conocimiento complementarias a las definidas previamente sin la necesidad de hacer cambios en el lenguaje.

Se conocen varias versiones de KIF para las cuales se intentó estandarizar sus especificaciones pero nunca se logró. El esfuerzo de ISO para una estandarización internacional comenzó en Junio de 2003 con la aprobación de una versión posteriormente desarrollada conocida como Common Logic[25] la cual logró cumplir con su proceso de estandarización para el año 2007.

Una variante llamada SUO-KIF[26] (Standard Upper Ontology Knowledge Interchange Format) es el lenguaje en el cual SUMO[27] (Suggested Upper Merged Ontology) son escritas.

Veamos un ejemplo de una definición usando KIF de un vehículo ferroviario diseñado para desplazarse por ferrocarriles.

```
(subclass RailVehicle LandVehicle)
  (documentation RailVehicle
```

[25] "Common Logic Standard". Disponible en: http://common-logic.org/#related. Consultado: Agosto 2009.

[26] "Standard Upper Ontology Knowledge Interchange Format". Disponible en: http://suo.ieee.org/SUO/KIF/suo-kif.html. Consultado: Agosto 2009

[27] Suggested Upper Merged Ontology (SUMO)". Disponible en: http://www.ontologyportal.org/ Consultado: Agosto 2009.

```
    "A Vehicle designed to move on &%Railways.")
  (=>  (instance ?X RailVehicle)
     (hasPurpose ?X
      (exists (?EV ?SURF)
       (and  (instance ?RAIL Railway)
           (instance ?EV Transportation)
           (holdsDuring (WhenFn ?EV)
                (meetsSpatially ?X ?RAIL))))))
```
[28]

2.3.2. OKBC(Open Knowledge Base Connectivity)[29]

Es un protocolo y un API para acceder a bases de conocimiento KB (Knowledge Bases) almacenadas en sistemas de representación del conocimiento KRSs(Knowledge Representation Systems). Su meta es servir como interfaz para los diferentes KRSs.

OKBC proporciona un conjunto de operaciones para una interfaz genérica que sirvan como base de los KRSs. La capa de interfaz permite un desarrollo en herramientas genéricas como browsers gráficos y editores con un alto grado de independencia de los software de KRSs.

Existen implementaciones de OKBC para varios lenguajes de programación como Java, C (Implementación del cliente solamente) y Lisp. Además proporciona acceso a KBs local o distribuido.

[28] "Knowledge Interchange Format". Disponible en: http://www.obitko.com/tutorials/ontologies-semantic-web/knowledge-interchange-format.html. Consultado: Abril 2010

[29] "Open Knowledge Base Connectivity". Disponible en: http://www.ai.sri.com/~okbc/ Consultado: Agosto 2009.

2.3.3. OCML(Operational Conceptual Modeling Language)[30]

Soporta la construcción de modelos de conocimiento por medio de significados de varios tipos de construcciones. Permite la especificación y manejo de funciones, relaciones, clases, instancias y reglas. También incluye mecanismos para definir ontologías y métodos para la solución de problemas.

Varios proyectos actualmente usan OCML en aplicaciones en las áreas de gestión de la información KM(Knowledge Management), Comercio electrónico y Sistemas basados en el conocimiento. OCML también soporta una gran librería de modelos reusables, proporcionando un recurso útil para la comunidad. Esta librería puede ser accedida desde el editor WebOnto[31].

Veamos un ejemplo de la definición de la relación "rango" utilizando OCML.

```
(def-relation RANGE (?f-r ?relation)
  :no-op (:iff-def(or
            (and(function ?f-r)
               (forall (?args ?result)
                     (=> (= (apply ?f-r ?args) ?result)
                        (holds ?relation ?result))))
            (and (binary-relation ?f-r)
                (forall (?x ?y)
                      (=> (holds ?f-r ?x ?y))
                         (holds ?relation ?y))))))
```
[32]

[30] "Operational Conceptual Modelling Language". Disponible en: http://technologies.kmi.open.ac.uk/ocml/ Consultado: Agosto 2009.
[31] http://projects.kmi.open.ac.uk/webonto/ Consultado: Agosto 2009.
[32] Motta, Enrico. "An Overview of the OCML Modelling Language". In Proceedings 8th Workshop on Knowledge Engineering Methods & Languages, 1998.

2.3.4 LOOM[33]

Es un lenguaje de descripciones lógicas para la representación del conocimiento, desarrollado por investigadores del grupo de inteligencia artificial de la Universidad del Sur de California. Su objetivo es desarrollar e implementar herramientas avanzadas de representación y razonamiento en inteligencia artificial.

Es un lenguaje para construir aplicaciones inteligentes. Proporciona un sistema de representación del conocimiento que es usado para proveer un soporte deductivo a una porción declarativa del lenguaje. El conocimiento declarativo en LOOM consta de definiciones, reglas, hechos y reglas predefinidas. Un motor deductivo llamado clasificador utiliza encadenamiento hacia adelante , unificación semántica y tecnologías orientadas a objetos de modo que exista una compilación de un conocimiento declarativo a una red diseñada eficientemente para soportar procesamiento deductivo de consultas en línea.

El sistema LOOM implementa una lógica basada en un comparador de patrones que facilita la producción de reglas y métodos de patrones dirigidos, soportando la definición de métodos orientados a objetos. El alto grado de integración entre los componentes declarativos o procedimentales de LOOM permite a los programadores utilizar paradigmas lógicos de programación, producción de reglas y paradigmas de programación orientada a objetos en una aplicación sencilla.

LOOM ha sido sucedido por PowerLoom Knowledge Representation and Reasoning System[34] que hereda en gran parte el funcionamiento de LOOM con nuevas características como un nuevo lenguaje de programación para su

[33] "Loom Project Home Page". Disponible en: http://www.isi.edu/isd/LOOM/LOOM-HOME.html. Consultado: Agosto 2009.

[34] "PowerLoom® Knowledge Representation & Reasoning System". Disponible en: http://www.isi.edu/isd/LOOM/PowerLoom/index.html. Consultado: Agosto 2009.

implementación llamado STELLA[35], el cual es fuertemente tipeado y puede ser traducido a Java, Lisp y C++.

Veamos algunos conceptos médicos utilizando LOOM:

```
(defconcept Body-Region
:is-primitive (:and Region
            (:some whole-location-of
             (:or Body-Part Tissue))
            (:some portion Organism))
 :implies (:and (:some connected Body-Region)
            (:some component (:or Body-System Body-Part))
            (:all near (:or Body-Region Body-Space Body-Part))
            (:all context-of (:or Biologic-Function Injury Poisoning))
            (:all crosses-through Body-Region))
:context anatomy)
(defconcept Fibromyalgia "UMLS-CUI C0016053"
   :is-primitive (:and
      Disorders-of-the-muscles-ligaments-fasciae-and-other-soft-tissues
      Muscle-functions-and-symptoms
      Myalgias/Myopathy
      Muscular-Diseases
      Rheumatic-Diseases
      Disease-or-Syndrome))[36]
```

[35] "STELLA - Lisp-style Symbolic Programming with Delivery in Common-Lisp, C++ and Java". Disponible en: http://www.isi.edu/isd/LOOM/Stella/ Consultado: Abril 2010.
[36] Pisanelli, Domenico M.; Gangemi, Aldo y Steve, Geri. "An Ontological Analysis of the UMLS MetathesaurusT", 1998

2.3.5. Frame Logic – FLogic[37]

Es un lenguaje ontológico para la representación del conocimiento. Se basa en aspectos estructurales de los lenguajes orientados a objetos y los lenguajes basados en frames (clases).

Entre sus características se incluye, identidad de objetos, objetos complejos, herencia, polimorfismo, métodos de consulta, encapsulación, entre otras.

```
/* schema facts */
Car:Vehicle.
Boat:Vehicle.
Bike:Vehicle.
Person[
   name => xsd#string;
   age => xsd#integer;
   friend =>> Person].
Vehicle[
   owner => Person;
   admissibleDriver =>> Person].
Car[
   passenger =>> Person;
   seats => xsd#integer].
/* facts */
peter:Person[
   name -> "Peter";
   age -> 17].
paul:Person[
   name -> "Paul";
```

[37] "Frame Logic". Disponible en: http://flora.sourceforge.net/aboutFlogic.php. Consultado: Agosto 2009.

```
   age -> 21;
   friend->peter].
mary:Person[
   name -> "Mary";
   age -> 17].
bike26:Bike[
   owner -> paul].
car74:Car[
   owner -> paul].
/* rules consisting of a rule head and a rule body */
FORALL X,Y X[friend->>Y] <- Y:Person[friend->>X].
FORALL X,Y X[admissibleDriver->>Y] <- X:Vehicle[owner->Y].
FORALL X,Y,Z X[admissibleDriver->>Z] <- X:Vehicle[owner->Y] AND
Y:Person[friend->>Z].
/* query */
FORALL X,Y <- X[admissibleDriver->>Y] AND X:Vehicle[owner->paul].
```
[38]

La primera sección de este ejemplo consta de un conjunto de "hechos esquema". El esquema representa de una manera orientada a objetos las clases y sus relaciones. Por ejemplo, el esquema define que cada persona tiene un nombre y una edad de tipo String y entero, respectivamente.

La segunda sección titulada "hechos", describe que algunos individuos pertenecen a la clase persona y consta de información sobre estos como su nombre y edad. También se define una relación entre los objetos, por ejemplo, Peter es amigo de Paul. De acuerdo con el paradigma orientado a objetos, las relaciones entre los objetos se representan con los métodos. Todos estos hechos

[38] GmbH, Ontoprise. “How to write F-Logic Programs”, 2008.

pueden ser considerados como la base de datos extendida de un programa en F-Logic.

2.3.6. SHOE (Simple HTML Ontology Extensions)[39]

Es un conjunto de extensiones de HTML diseñadas para darle a las páginas Web un significado semántico permitiendo información como clases, subclases y relaciones de propiedades.

Las etiquetas de SHOE se dividen en dos categorías. Primero, hay etiquetas para construir ontologías y segundo, hay etiquetas para anotación de documentos Web para suscribir una o más ontologías, declarar entidades de datos y hacer aserciones acerca de las entidades bajo las reglas definidas por las ontologías.

Su desarrollo ha terminado y éste en parte ha servido para la generación de estándares para los lenguajes que se describirán más adelante, tales como DAML + OIL y OWL.

Veamos un ejemplo de una ontología en SHOE. Como podemos ver es una extensión de HTML.

```
<HTML>
<HEAD>
   <!-- Aquí se indica que este documento esta hecho con SHOE 1.0-->     <META HTTP-
EQUIV="SHOE" CONTENT="VERSION=1.0">     <TITLE> Our CS Ontology
</TITLE> </HEAD> <BODY>
<!--Aquí declaramos el nombre y la versión de la ontología-->

<ONTOLOGY ID="cs-dept-ontology" VERSION="1.0">
<!--Aquí declaramos qué se está obteniendo prestado de otras ontologías-->
```

[39] "SHOE". Disponible en: http://www.cs.umd.edu/projects/plus/SHOE/ Consultado: Agosto 2009.

<USE-ONTOLOGY ID="base-ontology" VERSION="1.0" PREFIX="base"
URL="http://www.cs.umd.edu/projects/plus/SHOE/base.html">
<!-- Aquí exponemos nuestra jerarquía de categorías -->

<DEF-CATEGORY NAME="Organization" ISA="base.SHOEEntity"> <DEF-
CATEGORY NAME="Person" ISA="base.SHOEEntity"> <DEF-CATEGORY
NAME="Publication" ISA="base.SHOEEntity"> <DEF-CATEGORY
NAME="ResearchGroup" ISA="Organization"> <DEF-CATEGORY
NAME="Department" ISA="Organization"> <DEF-CATEGORY NAME="Worker"
ISA="Person"> <DEF-CATEGORY NAME="Faculty" ISA="Worker"> <DEF-
CATEGORY NAME="Assistant" ISA="Worker"> <DEF-CATEGORY
NAME="AdministrativeStaff" ISA="Worker"> <DEF-CATEGORY NAME="Student"
ISA="Person"> <DEF-CATEGORY NAME="PostDoc" ISA="Faculty"> <DEF-
CATEGORY NAME="Lecturer" ISA="Faculty"> <DEF-CATEGORY
NAME="Professor" ISA="Faculty"> <DEF-CATEGORY NAME="ResearchAssistant"
ISA="Assistant"> <DEF-CATEGORY NAME="TeachingAssistant" ISA="Assistant">
<DEF-CATEGORY NAME="GraduateStudent" ISA="Student"> <DEF-CATEGORY
NAME="UndergraduateStudent" ISA="Student"> <DEF-CATEGORY
NAME="Secretary" ISA="AdministrativeStaff"> <DEF-CATEGORY NAME="Chair"
ISA="AdministrativeStaff Professor">

<!-- ...Una manera opcional de decirlo <DEF-CATEGORY NAME="Chair"
ISA="AdministrativeStaff"> <DEF-CATEGORY NAME="Chair" ISA="Professor"> -->
<!-- Aquí exponemos las relaciones entre categorías -->
<DEF-RELATION NAME="advisor"> <DEF-ARG POS="1" TYPE="Student"> <DEF-
ARG POS="2" TYPE="Professor"> </DEF-RELATION> <DEF-RELATION
NAME="member"> <DEF-ARG POS="1" TYPE="Organization"> <DEF-ARG POS="2"
TYPE="Person"> </DEF-RELATION> <DEF-RELATION NAME="publicationAuthor">
<DEF-ARG POS="1" TYPE="Publication"> <DEF-ARG POS="2" TYPE="Person">
</DEF-RELATION>

<!-- Finalmente, exponemos nuestras otras relaciones -->
<DEF-RELATION NAME="publicationDate"> <DEF-ARG POS="1"
TYPE="Publication"> <DEF-ARG POS="2" TYPE=".DATE"> </DEF-RELATION>
<DEF-RELATION NAME="age"> <DEF-ARG POS="1" TYPE="Person"> <DEF-ARG
POS="2" TYPE=".NUMBER"> </DEF-RELATION> <DEF-RELATION
NAME="name"> <DEF-ARG POS="1" TYPE="base.SHOEEntity"> <DEF-ARG
POS="2" TYPE=".STRING"> </DEF-RELATION> <DEF-RELATION
NAME="tenured"> <DEF-ARG POS="1" TYPE="Professor"> <DEF-ARG POS="2"
TYPE=".TRUTH"> </DEF-RELATION>
</ONTOLOGY>

</BODY>
</HTML> [40]

2.3.7. XOL (Ontology Exchange Language)[41]

Es un lenguaje para el intercambio de ontologías por medio de XML. También fue diseñado para el intercambio de ontologías bioinformáticas, pero puede ser usado para ontologías de cualquier tipo.

Las definiciones de ontologías que XOL maneja incluyen esquemas de información (metadatos), como definiciones de clases de bases de datos de objetos y también incluye esquemas de no información, como definiciones de objetos de bases de datos de objetos.

La sintaxis de XOL se basa completamente en XML por sus ventajas ya conocidas y las semánticas de XOL se basan de OKBC-Lite. A continuación se presenta un ejemplo de una ontología de genealogía en XOL:

[40] Luke, Sean. "Creating Ontologies Using SHOE" Disponible en: http://www.cs.umd.edu/projects/plus/SHOE/ontologies.html. Consultado: Abril 2010
[41] "XOL Ontology Exchange Language". Disponible en: http://www.ai.sri.com/~pkarp/xol/ Consultado: Agosto 2009.

```
<?xml version="1.0" ?>
  <!DOCTYPE module SYSTEM "module.dtd">
  <module>
    <name>genealogy</name>
        <kb-type>ocelot-kb</kb-type>
        <package>user</package>

        <class>
          <name>person</name>
          <documentation>The class of all persons</documentation>
        </class>

        <class>
          <name>man</name>
          <documentation>The class of all persons whose sex is male
          </documentation>
          <subclass-of>person</subclass-of>
        </class>

        <class>
          <name>woman</name>
          <documentation>The class of all persons whose sex is female.
          </documentation>
          <subclass-of>person</subclass-of>
        </class>

        <slot>
          <name>year-of-birth</name>
          <documentation>An integer that represents the year the person was born.
          </documentation>
```

```
  <domain>person</domain>
  <slot-cardinality>1</slot-cardinality>
  <slot-numeric-min>1800</slot-numeric-min>
  <slot-value-type>integer</slot-value-type>
</slot>

<slot>
  <name>brothers</name>
  <documentation>The brothers of a person.</documentation>
  <domain>person</domain>
  <slot-value-type>man</slot-value-type>
</slot>

<slot>
  <name>citizenship</name>
  <documentation>Describes the citizenship status of a person.
  </documentation>
  <domain>person</domain>
  <slot-value-type>(set-of citizen resident-alien permanent-resident)
  </slot-value-type>
</slot>

<slot>
  <name>life-history</name>
  <documentation>A written history of the person's life.</documentation>
  <slot-value-type>string</slot-value-type>
</slot>

<slot>
  <name>father-of</name>
```

```
  <documentation>father-of(X,Y) holds when X is the father of Y.
  </documentation>
  <domain>man</domain>
  <slot-value-type>person</slot-value-type>
  <slot-inverse>father</slot-inverse>
</slot>

<slot>
  <name>has-father</name>
  <documentation>has-father(X,Y) holds when the father of X is Y.
  </documentation>
  <domain>person</domain>
  <slot-value-type>man</slot-value-type>
  <slot-inverse>father-of</slot-inverse>
</slot>

<individual>
  <name>John</name>
  <instance-of>man</instance-of>
  <slot-values>
     <name>year-of-birth</name>
     <value>1987</value>
  </slot-values>
  <slot-values>
    <name>citizenship</name>
    <value>permanent-resident</value>
  </slot-values>
  <slot-values>
    <name>has-father</name>
    <value>Carl</value>
```

```
  </slot-values>
 </individual>
  <individual>
  <name>Carl</name>
  <instance-of>man</instance-of>
  <slot-values>
    <name>year-of-birth</name>
    <value>1961</value>
  </slot-values>
  <slot-values>
    <name>father-of</name>
    <value>John</value>
  </slot-values>
  <slot-values>
    <name>life-history</name>
    <value>Carl worked hard all his life.</value>
  </slot-values>
 </individual>
</module>
```
[42]

2.3.8. OIL (Ontology Inference Layer, Ontology Interchange Language)[43]

Es una propuesta para la representación e inferencia de ontologías, la cual combina ampliamente primitivas de modelado de lenguajes basados en frames (clases) con semánticas y servicios de razonamiento proporcionados por descripciones lógicas (axiomas). Es compatible con RDFS e incluye semánticas precisas para describir significados.

[42] Karp, Peter D.; Chaudhri, Vinay K. y Thomere, Jerome. "XOL: An XML-Based Ontology Exchange Language", 1999
[43] "OIL". Disponible en: http://www.ontoknowledge.org/oil/ Consultado: Agosto 2009.

OIL representa un acercamiento al estándar de los lenguajes ontológicos. Podemos ver a OIL como una capa, pero cada capa adicional adhiere una nueva funcionalidad y complejidad a la capa anterior. Esto hace que los agentes(humanos o máquinas) que solo pueden procesar una capa baja puedan todavía entender ontologías que son expresadas en cualquiera de las capas superiores.

Veamos la definición de la "vida salvaje africana" utilizando OIL.

class-def animal
class-def plant subclass-of NOT animal
class-def tree subclass-of plant

class-def deWned herbivore subclass-of animal
NOT carnivore
slot-constraint eats value-type plant
OR (slot-constraint is-part-of has- value plant)
class-def giraVe subclass-of herbivore
slot-constraint eats value-type leaf [44]

2.3.9. DAML + OIL[45]

Es el lenguaje sucesor de DAML(Darpa Agent Markup Language)[46] y OIL en conjunto, combinando características de ambos. El objetivo de DAML era encontrar un lenguaje y herramientas para facilitar el concepto de Web Semántica.

[44] Pulido, J. R. G.; Ruiz, M. A. G.; Herrera, R.; Cabello, E.; Legrand, S. y Elliman, D. "Ontology languages for the semantic web: A never completely updated review". Knowledge-Based Systems, Volume 19, Issue 7, 2006, p. 489-497.
[45] "DAML+OIL (March 2001)". Disponible en: http://www.daml.org/2001/03/daml+oil-index. Consultado: Agosto 2009.
[46] "The DARPA Agent Markup Language Homepage". Disponible en: http://www.daml.org/ Consultado: Agosto 2009

Un conjunto de sentencias DAML por sí mismas (y la especificación DAML) permiten concluir otra sentencia DAML mientras que un conjunto de declaraciones XML por sí mismas (y la especificación XML) no permite concluir otras sentencias XML. Para usar XML y generar nuevos datos, se necesita conocimiento incluido en algún código de procedimiento como el que está explícitamente establecido en DAML.

DAML ofrece propiedades estándar tales como equivalencia, o propiedades particulares únicas. De esta forma, el conocimiento podría ser aplicado dinámicamente para encontrar una respuesta, mediante la utilización de un lenguaje expresivo enriquecido.

Veamos el ejemplo de restringir la referencia de un elemento en un archivo utilizando DAML + OIL.

```
<daml:Class rdf:ID="Adult">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Person"/>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#age"/>
          <daml:hasClass     rdf:resource="http://www.daml.org/2001/03/daml+oil-ex-
dt#over17"/>
    </daml:Restriction>
  </daml:intersectionOf>
</daml:Class>
<daml:Class rdf:ID="Senior">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Person"/>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#age"/>
      <daml:hasClass rdf:resource="http://www.daml.org/2001/03/daml+oil-ex-dt#over59"/>
```

```
  </daml:Restriction>
 </daml:intersectionOf>
</daml:Class>
```
[47]

2.3.10. OML/CKML(Conceptual Knowledge Markup Language) [48,49]

CKML como extensión de OML es un lenguaje basado en grafos conceptuales a diferencia de todos los otros lenguajes mencionados. CKML usa XML para su sintaxis.

CKML proporciona un framework de conocimiento conceptual para la representación de información distribuida. Se basa en el principio que incorpora métodos e instrumentos de procesamiento de conocimiento conceptual que ayuda a las personas en sus pensamientos racionales, juicios, actuaciones y promueve discusiones críticas. Además incorpora nuevas técnicas de flujos de información y diseño lógico de sistemas distribuidos.

OML tiene dos partes de lenguaje: ontologías como un contenedor de tipos de ontologías y colecciones como un contenedor de instancias que asocian las ontologías.

CKML además incluye los elementos básicos de un flujo de información como son: Clasificaciones, Infomorfismos, Teorías, Interpretaciones y Lógicas Locales.

Veamos una representación de la expresión: $\exists x: Ball(color(Red) \wedge chrc(x, Red))$ utilizando CKML.

[47] "Annotated DAML+OIL (March 2001) Ontology Markup". Disponible en: http://www.daml.org/2001/03/daml+oil-walkthru.html. Consultado: Abril 2010.
[48] "OML/CKML Grammar". Disponible en: http://www.ontologos.org/OML/CKML-Grammar.html. Consultado: Agosto 2009.
[49] "Conceptual Knowledge Markup Language". Disponible en: http://www.ontologos.org/CKML/CKML%200.2.html. Consultado: Agosto 2009.

<Ontology>

. . .

<Type.Object name="Color"/>

<Type.Object name="Red"/>

. . .

<classification instance="Red" type="Color"/>

. . .

<Type.Object name="Ball"/>

<Type.BinaryRelation name="chrc" source.Type="Ball"

target.Type="Color"/>

</Ontology>

/* specific style */

<Collection>

. . .

<Ball>

<chrc target.Instance="Red"/>

</Ball>

. . .

</Collection> [50]

2.3.11. RDF(Resource Description Framework) [51]

RDF es un lenguaje para especificar metadatos, diferente a XML que es un lenguaje para modelar datos.

Cuando se tienen grandes volúmenes de datos, XML no es lo suficientemente escalable pues el orden de los elementos que maneja no es natural y de alguna manera el mantenimiento de los datos termina siendo extremadamente difícil y

[50] Kent, R. E. "Conceptual Knowledge Markup Language: The Central Core". In Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management, 1999.

[51] "Resource Description Framework (RDF)". Disponible en: http://www.w3.org/RDF/ Consultado: Agosto 2009.

costoso. Con RDF no ocurre esto pues cuenta con la interoperabilidad entre aplicaciones que intercambian información comprensible por la página web, para proporcionar una infraestructura que soporte actividades de metadatos.

La definición de RDF está basada en las recomendaciones del W3C: definición, conceptos, sintaxis, semántica, vocabulario (schema) y casos de prueba (test cases). A continuación se muestra en resumen cada una de estas recomendaciones o especificaciones:

Definición RDF[52]: Ofrece los conceptos y conocimientos básicos para entender y usar RDF, la descripción de la sintaxis XML. Describe como definir los vocabularios RDF usando el mismo lenguaje de descripción del vocabulario RDF. Además ofrece una breve introducción al uso y aplicaciones de RDF. También muestra el contenido de otros documentos y especificaciones RDF no menos importantes para su total entendimiento.

Conceptos y sintaxis abstracta[53]: Define la sintaxis abstracta en la que funciona RDF. Explica para qué enlazar una sintaxis a una semántica formal. Además incluye una discusión sobre objetivos del diseño, conceptos clave, tipos de datos, normalización de caracteres y manejo de referencias URI.

Especificación de sintaxis de RDF/XML[54]: Define la sintaxis XML para RDF llamada RDF/XML en términos de namespaces, XML Information Set y XML Base. La gramática formal para la sintaxis es anotada con acciones generadas en tripletas del grafo RDF que está definido en[55]. Las tripletas son escritas usando el formato de serialización de grafos RDF N-Triples que hace posible el mapeo para un almacenamiento más preciso y procesable para la máquina. Los mapeos son

[52] "RDF Primer". Disponible en: http://www.w3.org/TR/rdf-primer/ Consultado: Diciembre 2009.
[53] "Resource Description Framework (RDF): Concepts and Abstract Syntax". Disponible en: http://www.w3.org/TR/rdf-concepts/ Consultado: Diciembre 2009.
[54] "RDF/XML Syntax Specification (Revised)". Disponible en: http://www.w3.org/TR/rdf-syntax-grammar/ Consultado: Diciembre 2009.
[55] "Resource Description Framework (RDF): Concepts and Abstract Syntax". Disponible en: http://www.w3.org/TR/rdf-concepts/ Consultado: Diciembre 2009.

guardados en forma de casos de prueba, siendo recolectados y publicados en casos de prueba RDF.

Semántica RDF[56]: Ofrece una semántica precisa y un completo sistema de reglas de inferencia para RDF y RDF Schema.

Descripción del vocabulario del lenguaje RDF Schema[57]: Describe como se debe usar RDF para describir vocabularios RDF. Describe un vocabulario para este propósito y describe otros vocabularios construidos inicialmente en RDF especificados en RDF Model and Syntax.

Casos de prueba RDF[58]: Describe los RDF Test Cases que hagan cumplir las reglas de la gramática, sintaxis, cadena, N-Triples, en general todo lo relacionado a cómo debe funcionar RDF.

De esta forma se describe como está compuesto RDF. Para una explicación más amplia nos remitimos a Definición RDF para un mejor entendimiento.

RDF está particularmente diseñado para representar metadatos sobre recursos Web tales como título, autor, modificaciones de fechas, fechas de creación, de borrado, derechos de autor, licencias y en general todo lo relacionado con el documento. Pero esta definición puede ser extendida a otros recursos Web de forma que sirva para definir o describir objetos comunes con información tal como especificaciones, precios, disponibilidad, cantidad, etc.

No sólo mostrarle información al usuario es el objetivo de una página Web, es aún más importante que el sistema reconozca toda esa información y sea capaz de procesarla para las necesidades del negocio. Toda esta información debe ser

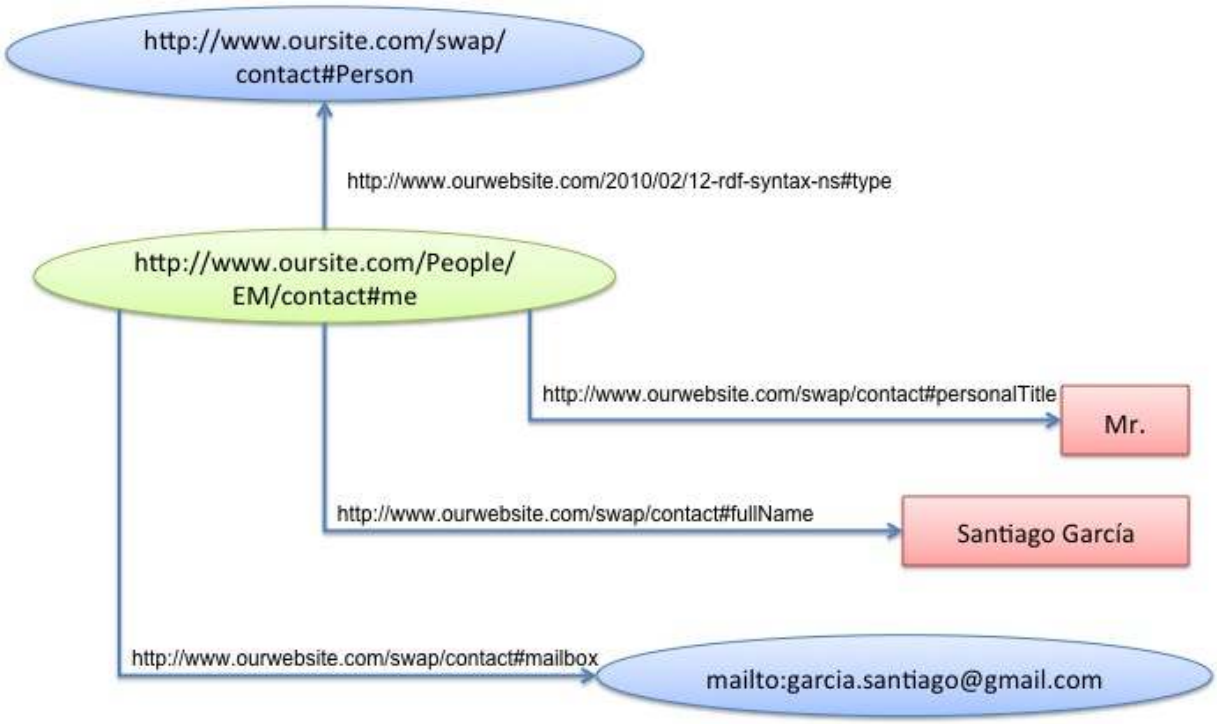
[56] RDF Semantics". Disponible en: http://www.w3.org/TR/rdf-mt/ Consultado: Diciembre 2009.
[57] "RDF Vocabulary Description Language 1.0: RDF Schema". Disponible en: http://www.w3.org/TR/rdf-schema/ Consultado: Diciembre 2009.
[58] "RDF Test Cases". Disponible en: http://www.w3.org/TR/rdf-testcases/ Consultado: Diciembre 2009.

procesada por la aplicación sin que pierda sentido. RDF provee un framework común que permite expresar toda esta información de manera que pueda ser intercambiada entre diferentes aplicaciones. Esto implica que la información esté disponible para otras aplicaciones diferentes a la programada originalmente.

RDF se basa en la idea de identificar cosas usando identificadores Web, llamados URIs (Uniform Resource Identifiers) y describir los recursos en términos de propiedades simples y propiedades con valores. Esto le permite a RDF representar sentencias simples con grafos.

La sentencia "Hay una persona identificada con http://www.ousite.com/People/EM/contact#me con el nombre Santiago García y cuyo correo electrónico es garcia.santiago@gmail.com, y cuyo título es Mr." puede ser representada por el siguiente grafo.

Figura 1. Grafo RDF



En la Figura 1 podemos ver como RDF usa URIs para la identificación de objetos:

- Individuos: por ejemplo, Santiago García se identifica con http://www.oursite.com/People/EM/contact#me
- Tipos de Cosas: por ejemplo, Persona se identifica con http://www.oursite.com/swap/contact#Person
- Propiedades o atributos de las cosas: por ejemplo, Mailbox se identifica con http://www.oursite.com/swap/contact#mailbox
- Valores para las propiedades: en este caso se refieren a mailto:garcia.santiago@gmail.com, Santiago Garcia y Mr. Se puede ver como RDF también hace uso de Strings y otros tipos de datos como enteros y fechas.

El grafo anterior puede ser representado con XML usando RDF/XML de la siguiente manera:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.ourwebsite.com/2010/02/12-rdf-syntax-ns#"
   xmlns:contact="http://www.ourwebsite.com/swap/contact#">
<contact:Person rdf:about="http://www.ourwebsite.com/People/EM/contact#me">
   <contact:fullName>Santiago Garcia</contact:fullName>
   <contact:mailbox rdf:resource="mailto:garcia.santiago@gmail.com"/>
   <contact:personalTitle>Mr.</contact:personalTitle>
</contact:Person>
</rdf:RDF> [38]
```

Nótese que RDF/XML también contiene URIs y propiedades como mailbox y fullName con sus respectivos valores dentro de las etiquetas.

En este momento podríamos decir que RDF/XML es lo mismo que XML o HTML, pero la diferencia se encuentra en que en este momento estamos procesando entidades comunes de información en la Web. Con las URIs nos podemos referir a cualquier cosa identificable incluyendo objetos como carros, libros, negocios, personas, eventos, viajes, etc. que aún no son identificables con hipertexto y que no tendría ningún sentido. La Web se potencializa con RDF y sus propiedades que por sí mismas tienen URIs para precisamente identificar las relaciones que existen entre los objetos enlazados.

2.3.12 OWL(Web Ontology Language)

El lenguaje de ontologías para la Web (OWL) está diseñado principalmente para el manejo de información entre diferentes aplicaciones por medio de ontologías. En realidad OWL es una extensión del lenguaje RDF y como lo habíamos descrito anteriormente estos lenguajes están creados de cara al procesamiento de datos

entre programas o aplicaciones, contrariamente a los lenguajes creados exclusivamente para mostrar información al usuario.

OWL tiene más funcionalidades para expresar el significado y semántica que XML, RDF y RDFS (RDF Schema), además de proveer un mayor vocabulario de semánticas formales. OWL tiene ventaja sobre estos lenguajes en el sentido de que puede representar contenido en la Web y este a su vez es interpretable por la máquina. OWL es una revisión del lenguaje DAML+OIL que incorpora lecciones aprendidas desde el diseño y la aplicación de este lenguaje.

OWL se describe en un conjunto de documentos, cada uno con diferentes alcances. Para el mayor entendimiento de OWL, ésta es la lista de los documentos:

- Visión general de OWL[59]: Ofrece una lista de propiedades y principales usos de forma introductoria.
- Guía de OWL[60]: Muestra el uso de OWL con un ejemplo extenso y además provee un glosario[61] con la terminología usada en estos documentos.
- Referencia de OWL[62]: Descripción sistemática y resumida del uso de las primitivas de modelado de OWL.
- Semántica y sintaxis abstracta de OWL[63]: Descripción formal y final del lenguaje.

[59] "OWL Web Ontology Language Overview". Disponible en: http://www.w3.org/TR/owl-features/ Consultado: Diciembre 2009.
[60] "OWL Web Ontology Language Guide". Disponible en: http://www.w3.org/TR/owl-guide/ Consultado: Diciembre 2009.
[61] Disponible en: http://www.w3.org/TR/owl-guide/#OWLGlossary. Consultado: Diciembre 2009.
[62] Bechhofer, Sean; van Harmelen, Frank; Hendler, Jim; Horrocks, Ian; McGuinness, Deborah L.; Patel-Schneider, Peter F. y Stein, Lynn Andrea. "OWL Web Ontology Language Reference". Disponible en: http://www.w3.org/TR/owl-ref/ Consultado: Diciembre 2009.
[63] "OWL Web Ontology Language Semantics and Abstract Syntax". Disponible en: http://www.w3.org/TR/owl-semantics/ Consultado: Diciembre 2009.

- Casos de prueba de OWL[64]: contiene un amplio conjunto de casos de prueba para el lenguaje.
- Casos de uso y requerimientos de OWL[65]: Contiene un amplio conjunto de casos de uso y requerimientos de OWL.

El lenguaje OWL tiene tres sub-lenguajes de expresión incremental diseñados para el uso de comunidades específicas de desarrolladores o usuarios, según sea el uso y el nivel de expresividad requerido.

Figura 2. OWL y sus tres sub-lenguajes

[64] "OWL Web Ontology Language Test Cases". Disponible en: http://www.w3.org/TR/owl-test/ Consultado: Diciembre 2009.
[65] "OWL Web Ontology Language Use Cases and Requirements". Disponible en: http://www.w3.org/TR/webont-req/ Consultado: Diciembre 2009.

OWL Lite: Para los usuarios que necesitan clasificaciones jerárquicas y restricciones simples. Por ejemplo, soportar restricciones cardinales, éste solo permite valores cardinales de 0 ó 1. Debido a su simpleza es mas fácil proveer una herramienta de soporte. Ofrece una rápida ruta de migración para tesauro y otras taxonomías. El nivel de complejidad es menor que el de OWL DL y OWL Full.

OWL DL: Para los usuarios que necesitan de una alta expresividad manteniendo una completa computacionalidad y resolubilidad, es decir, para todas las funciones se garantiza que sean computables y que terminen en un tiempo finito. Incluye todas las construcciones de OWL pero éstas solo pueden ser usadas bajo ciertas restricciones (por ejemplo, mientras una clase puede ser subclase de muchas clases, una clase no puede ser instancia de otra clase). Se llama OWL DL en correspondencia a *description logics*, un campo de investigación que ha estudiado la lógica para la fundación formal de OWL. El nivel de complejidad es menor que el de OWL Full.

OWL Full: Para los usuarios que necesitan la máxima expresividad y máximo uso de la sintaxis, pero no garantiza computacionalidad. Por ejemplo, una clase puede ser tratada simultáneamente como una colección de individuos y como un individuo por derecho propio. Es poco probable que algún software pueda soportar el razonamiento completo para cada propiedad de OWL Full.

Los desarrolladores de ontologías deben considerar cuál sublenguaje se adapta mejor a sus necesidades. La escogencia entre OWL Lite y OWL DL depende del grado de mayor o menor expresividad que quiera darse. La escogencia entre OWL DL y OWL Full principalmente depende del grado de extensión que los usuarios requieran para el modelado de metadatos, por ejemplo, definir clases de clases y adicionar propiedades a las clases. Se debe tener en cuenta que el soporte para

OWL Full es menor dado que no existe una implementación actual completa de OWL Full.

OWL Full puede ser visto como una extensión de RDF, mientras que OWL Lite y OWL DL pueden ser vistos como extensiones de una vista restringida de RDF. Cada documento OWL(Lite, DL, Full) es un documento RDF y cada documento RDF es un documento OWL Full, pero sólo algunos documento RDF pueden ser vistos como documentos válidos de OWL Lite ó OWL DL. Entre otras, cada URI que es usada como nombre de clase debe ser explícitamente acertado a un tipo owl:Class, similarmente para las propiedades. Cada individuo debe por lo menos pertenecer a una clase (incluso si es solo owl:Thing), las URIs usadas para las clases, propiedades e individuos deben ser mutuamente disyuntas. Para mayor información referirse al apéndice E de referencia de OWL[66].

A continuación se presenta una ontología realizada en OWL basado en RDF sobre XML.

IPROnto Intellectual Property Rights Ontology (Ontología sobre derechos de autor)[67]:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rss="http://purl.org/rss/1.0/"
   xmlns:jms="http://jena.hpl.hp.com/2003/08/jms#"
   xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
   xmlns="http://dmag.upf.es/ontologies/2003/12/ipronto.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
```

[66] Bechhofer, Sean et al. Op. Cit.
[67] Disponible en: http://dmag.upf.es/ontologies/2003/12/ipronto.owl. Consultado: Diciembre 2009.

```
   xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xml:base="http://dmag.upf.es/ontologies/2003/12/ipronto.owl">
<owl:Ontology rdf:about="">
   <dc:creator>Distributed Multimedia Applications Group
      (http://dmag.upf.es)</dc:creator>
   <dc:title>Intellectual Property Rights Ontology</dc:title>
   <dc:description>An ontology to model Intellectual Property Rights suited for
      Digital Rights Management</dc:description>
   <dc:date>2001-12-08</dc:date>
   <dc:subject>Intellectual Property Rights, Digital Rights, IPR, DRM,
      Ontology</dc:subject>
</owl:Ontology>

<owl:Class rdf:ID="LegalEntity">
<owl:Class rdf:ID="Abstraction">
<owl:Class rdf:ID="CiteRight">
<owl:Class rdf:ID="Reproduce">
<owl:Class rdf:ID="IPRLicense">
<owl:Class rdf:ID="CorporateLegalEntity">
<owl:Class rdf:ID="GeographicArea">
<owl:Class rdf:ID="Creator">
<owl:Class rdf:ID="DistributionLicense">

(.....................)

</rdf:RDF>
```

La ontología inicia con la declaración de namespaces XML con la etiqueta rdf:RDF y muestra las URIs que identifican dicha ontología en la Web.

Después de los namespaces comienzan las declaraciones OWL agrupadas bajo la etiqueta owl:Ontology. En esta sección en orden de ocurrencia se encuentra en primer lugar los metadatos OWL. Posteriormente aparecen los elementos básicos de OWL tales como clases, subclases, individuos, propiedades y relaciones.

```
<owl:Class rdf:ID="LegalEntity">
   <rdfs:comment>An entity possessing the capacity in law to exercise or enjoy
      an intellectual property right.</rdfs:comment>
   <rdfs:subClassOf>
      <owl:Class rdf:about="#LegalConcept" />
   </rdfs:subClassOf>
   <owl:equivalentClass>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#NaturalLegalEntity" />
            <owl:Class rdf:about="#CorporateLegalEntity" />
         </owl:unionOf>
      </owl:Class>
   </owl:equivalentClass>
   <rdfs:label>Person</rdfs:label>
</owl:Class>
```

Se puede ver como se define una clase, así como sus subclases, clases equivalentes y relaciones.

```
<owl:Class rdf:ID="GeographicArea">
```

```
    <rdfs:comment>A geographic location, generally having definite boundaries.
      Note that this differs from its immediate superclass Place in that a
      GeographicArea is a Place of significant size.</rdfs:comment>
    <rdfs:label>GeographicArea</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Place" />
    </rdfs:subClassOf>
</owl:Class>
```

Esta es la clase "GeographicArea" con sus respectivos comentarios donde también se define que es una subclase de la clase "Place".

3. TÉCNICAS DE RECUPERACIÓN DE INFORMACIÓN BASADAS EN ONTOLOGÍAS

Luego de contextualizarnos con el marco teórico vamos a centrarnos en las técnicas de recuperación de información basadas en ontologías que actualmente se están utilizando. La implementación más directa la podemos observar en los motores de búsqueda y metadatos, especializados en manejar las cualidades de la Web semántica para brindar una información de mayor calidad involucrando ontologías y documentos semánticos. Se analizarán los diferentes proyectos y motores que se han construido, además de las técnicas que estos actualmente usan.

Las técnicas de recuperación de información las podemos clasificar dependiendo de las cualidades que utilizan para garantizar unos resultados más confiables y precisos en la información suministrada al usuario. En las siguientes secciones se describirán cada una de las técnicas y las aproximaciones que se han dado en los motores de búsqueda basados en ontologías actuales.

3.1. TÉCNICAS DE PERFILES DE USUARIOS Y ONTOLOGÍAS PERSONALIZADAS

Los perfiles y modelos de usuario son un complemento a las ontologías. Podemos ver que en el área de la recuperación de información ambos juegan un papel importante que busca brindarle al usuario esencialmente precisión y relevancia en la información que solicita.

La idea de tener un motor de búsqueda que conozca los intereses del usuario constituye el elemento diferenciador frente a los esquemas que actualmente estamos utilizando para encontrar la información en una Web que tiene un crecimiento exponencial. Podemos decir que el establecimiento de un modelo de usuario es la clave para convertirse en una búsqueda personalizada, además un

modelo de usuario no es una descripción general de un usuario en específico, pero sí un tipo de orientación algorítmica, teniendo una estructura de datos específica y descripciones de usuarios formalizadas que permiten cada vez conocer más los intereses del usuario.

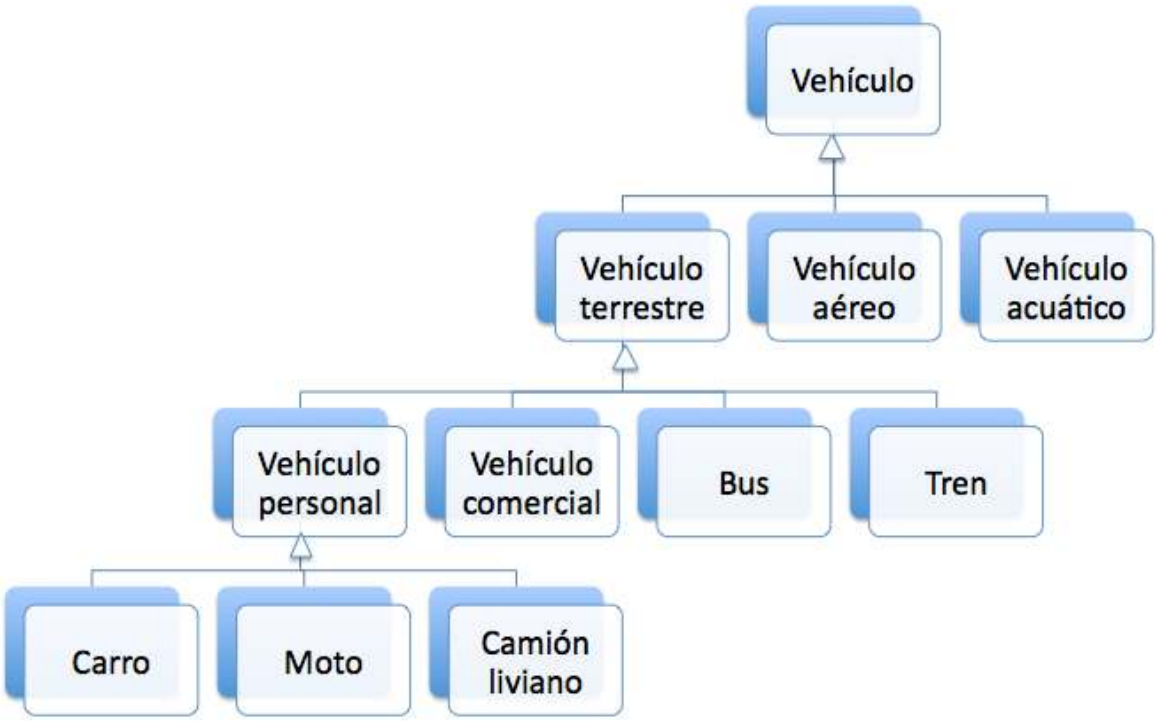
3.1.1. Búsqueda personalizada basada en ontología[68]

Se empezará exponiendo uno de los primeros acercamientos a la búsqueda personalizada basada en ontologías. Para esto se debe tener en cuenta la forma de recolección del contenido, los perfiles de usuario y el proceso de búsqueda ontológica en general. Veamos entonces a partir de estos temas como se construye una búsqueda personalizada basada en ontologías.

3.1.1.1. Determinación del contenido de los documentos. Los intereses de usuario se infieren de las visitas que los usuarios hacen a las páginas Web, para esto es necesario determinar el contenido de las páginas visitadas. Esto se hace usando ontologías que se basan en una jerarquía de datos de navegación públicamente accesibles. Para este caso bastante general se tienen pocas relaciones semánticas y éstas se dan principalmente por relaciones de especialización, es decir super componente o subcomponente.

[68] Pretschner, Alexander y Gauch, Susan. "Ontology Based Personalized Search", Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence, 1999.

Figura 3. Representación de una ontología



En la figura 3 podemos ver un ejemplo de las relaciones de especialización. Son las mismas utilizadas en las relaciones de la ontología propuesta.

Se cuenta entonces con una ontología, con aproximadamente 4400 nodos y cada nodo se asocia a un conjunto de documentos que representan el contenido del nodo. Por cada conjunto de documentos de un nodo se crea un super documento. Tanto para el conjunto de documentos como para el superdocumento se tienen palabras claves con un determinado peso asignado por la frecuencia de los términos o la frecuencia del documento. Para cada página visitada se calcula un vector de palabras claves (peso). Este peso es comparado con los demás pesos que se encuentran en los nodos para calcular similitudes. Los nodos con el máximo número de vectores que coinciden se asumen como los que están más relacionados con el contenido de la página visitada.

3.1.1.2. Perfiles de usuario. Se definen por la aproximación de los intereses del usuario y se basan principalmente en una jerarquía estructurada de usuarios, en una generación automática sin retroalimentación explícita y un mecanismo dinámico para un proceso de aprendizaje continuo.

Los perfiles de usuario son creados de acuerdo a la navegación de éstos en la página Web, las retroalimentaciones manuales no son tenidas en cuenta debido a que en algunos casos pueden generar molestia en los usuarios y no son transparentes en el flujo de trabajo del usuario.

3.1.1.3. Proceso de asignación de pesos a las relaciones de términos. La generación de perfil y adaptación se basan en que las páginas que se encuentran en la caché del Web Browser son continuamente clasificadas en categorías y áreas. Las categorías con más peso son combinadas con el tiempo que el usuario pasa en la página y la cantidad de contenido de la página. El peso asignado sólo puede subir y no se tiene en cuenta cuando a los usuarios en realidad no les gusta alguna página, en este caso la cierran automáticamente después de ver su contenido y no hay calificación negativa.

Después de varias combinaciones de las variables tiempo, tamaño de la página y discriminantes de la búsqueda se tiene lo siguiente:

$\gamma(d,c_i)$ es la fuerza entre el la combinación del contenido del documento d y la categoría c_i. El resultado es la caracterización de la página.

$\Delta_{\iota}(c_i)$ es el ajuste del interés ι en una categoría c_i.

$\Delta_{\iota}(c_i)$

Definidas las relaciones anteriores en términos de convergencia y mejora de resultados de búsqueda se tienen las siguientes funciones:

$$\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i) \quad (1)$$

$$\Delta\iota(c_i) = \log \frac{time}{\log \log length} \cdot \gamma(d, c_i) \quad (2)$$

En las ecuaciones 1 y 2 la longitud o tamaño de la página es despreciable. En términos de convergencia de perfiles después de varios experimentos se prueba que funcionan muy bien. Se debe resaltar que el tamaño de la página es despreciable dado que el comportamiento convergente de las funciones anteriores indica que los usuarios simplemente escogen de un vistazo que les interesa y lo leen sin importar si es largo o corto, la importancia se traduce en la relevancia para el usuario. El método entonces da mayor relevancia a un contenido importante con un tiempo alto consumido en su lectura.

Los experimentos realizados muestran muy buenos resultados y es la base de otras técnicas de expansión de consultas, ranking y filtros. Todas estas se cubrirán en este documento. Esta parte sólo es un acercamiento general a una arquitectura de una búsqueda personalizada basada en ontologías.

3.1.2. Motor de búsqueda personalizado basado en Ontologías[69]

Este motor de búsqueda cuenta con la ayuda de los conocimientos de interés personal de los usuarios, lleva a cabo el análisis semántico para compartir el banco de conocimiento de las ontologías, filtra los resultados de las búsquedas y conoce las consultas personalizadas de los diferentes usuarios para obtener diferente información para la misma consultada.

[69] Li Yong y Li Guan-yu. "Research and realization of personalized search engine based on Ontology". Proceedings of the IFIP International Conference on Network and Parallel Computing Workshops, 2007.

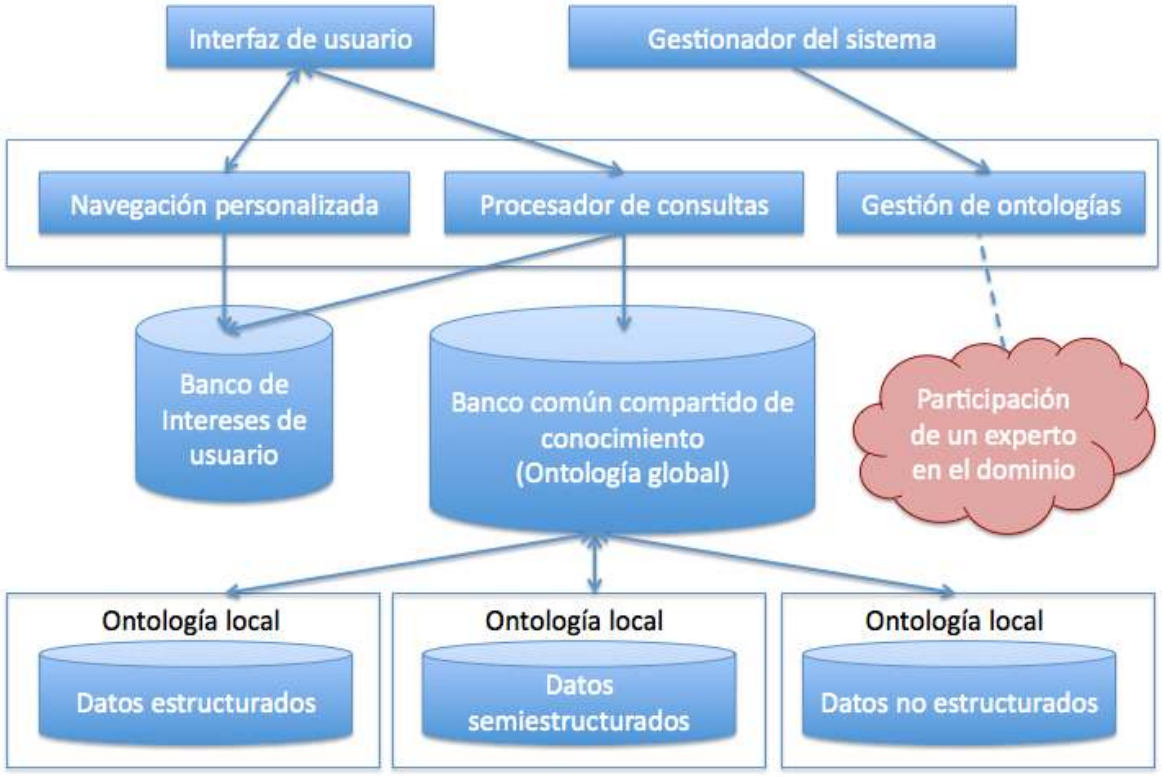
Para poder hablar de información personalizada por usuario se define lo que llamamos modelo de usuario. El modelo de interés de los usuarios apunta a un modelo en el campo de información específica donde su definición la podemos representar con la siguiente tripleta.

$$ModeloDeUsuario = \{personalI, personalR, personalW\} \quad (3)$$

Donde $personalI$ representa la información personal del usuario, incluye el nombre de usuario, sexo, edad, antecedentes educativos, antecedentes profesionales, etc. $personalR$ representa las palabras de interés para el usuario y $personalW$ representa el grado de interés por dichas palabras. A continuación veremos un algoritmo que aprende del modelo de usuario:

```
SI (usuario logueado) {
  SI (no existe personalI) {
    Inicializar el banco de interés del usuario;
    Recoger la información de interés del usuario;
    Modificar el banco de interés del usuario;
  }
}
SI (usuario consulta) {
   Obtener solicitud de consulta;
   Construir expresión de consulta mediante la interacción entre
   el usuario y el sistema;
   Insertar la solicitud de consultar en el banco de interés del
   usuario;
   Calcular el grado de interés del usuario para cierto
   concepto;
   Obtener el log de visita del usuario;
   Mantener el banco de interés del usuario;
}
```

Figura 4. Arquitectura utilizada para las búsquedas personalizadas basadas en ontologías



La figura 4 anterior nos muestra la arquitectura utilizada, el algoritmo que se acaba de describir hace parte del procesador de consultas de la arquitectura presentada.

Luego de tener definido el modelo de usuario en el motor de búsqueda basado en ontología se realizarían las siguientes actividades:

- Filtro personalizado para las consultas solicitadas por los usuarios.
- El módulo de interés del usuario define el interés del usuario.
- Los agentes de búsqueda toman el interés del usuario para llevar a cabo la búsqueda.
- Los resultados son filtrados.

- Los resultados son ordenados dependiendo del grado de interés, organización y retroalimentación dada por el usuario.
- El usuario y el sistema interactúan para la actualización del banco de conocimiento y el banco de información. Al mismo tiempo los sistemas se esfuerzan por recuperar información que el usuario determina, provee y ayuda a completar sus intereses del usuario.
- El sistema recomendará información de acuerdo al interés de cada usuario a través de e-mail, mensajes, etc. Esto significa cambiar la afirmación: "Las personas buscan información" por "La información busca a las personas".

3.1.3. Servicio de búsqueda personalizada basado en Ontología personalizada[70]

Este servicio está basado en los intereses del usuario y le brinda los resultados más útiles. Por consiguiente se presenta la siguiente definición de ontología personalizada:

$$O=(C,R,I,A,F) \quad (4)$$

Donde C es el conjunto de conceptos de la ontología, R es el conjunto de atributos del concepto, I es el conjunto de instancias del concepto, A es el conjunto de tácticas de la ontología, el cual es usado para describir las relaciones lógicas entre entidades. F es el conjunto de funciones, el cual mapea los conceptos y los atributos al peso de interés de usuario. F se define a continuación:

$$F:O->(W_{rf},N_{sc}) \quad (5)$$

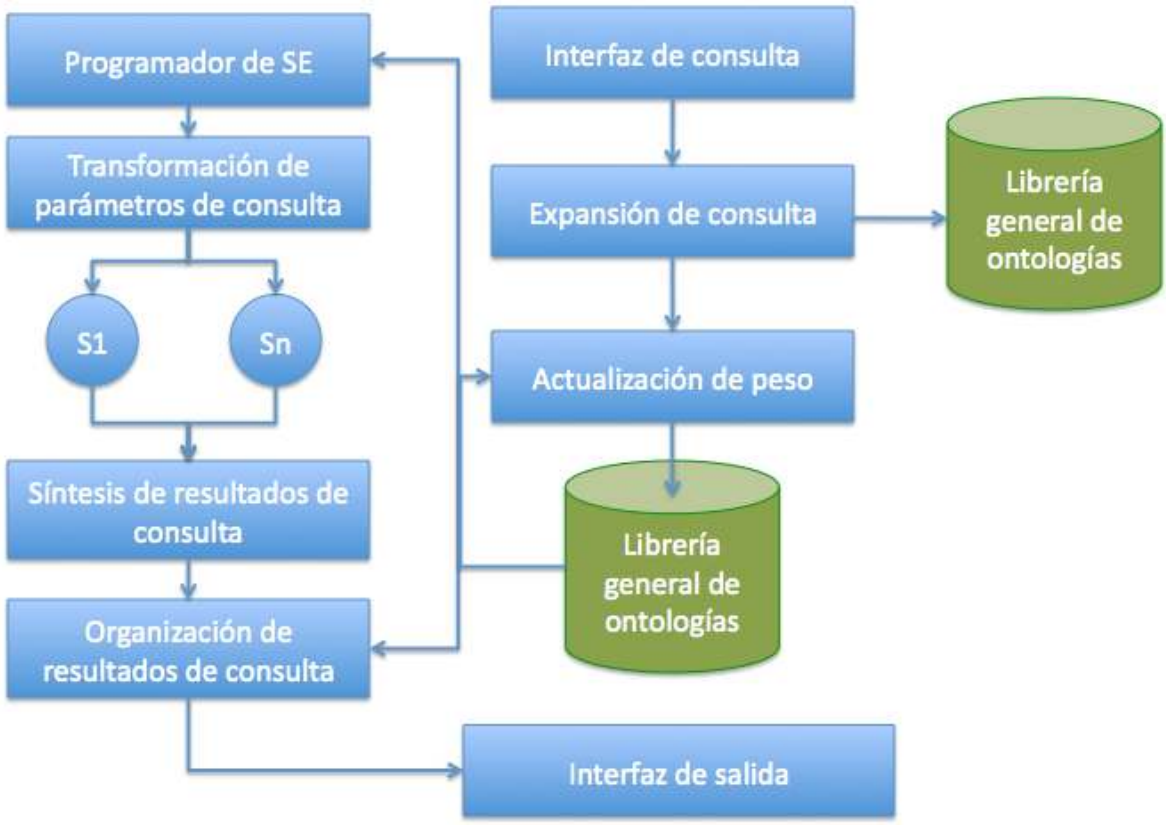
[70] Zhijun Zhang y Yunqing Li. "Research and Implementation of the Personalized Meta Search Engine based on Ontology", Web Intelligence and Agent Systems, Volume 1 , Issue 3-4, 2003, p. 219 - 234.

O es la librería ontológica, W_{rf} es el peso de interés del usuario y N_{se} es un miembro de los motores de búsqueda independientes. Para actualizar la librería de ontología personalizada mientras los usuarios utilizan el servicio se presentan dos factores que tienen un impacto significativo en la actualización de la misma. Éstos se presentan como la conclusión de observar el comportamiento de búsqueda de los usuarios para medir el sus interés acerca de algún concepto. Primero, entre más veces es buscado el concepto más interesante es para los usuarios. Segundo, entre más veces se le hace clic al concepto más exacto es para los usuarios. En consecuencia, se define el peso de interés de usuario como:

$$W_{rf}(c)_{i+1} = W_{rf}(c)_i * \beta^{\,b} + W(c)_i \qquad (6)$$

Donde $W_{rf}(c)_i$ y $W_{rf}(c)_{i+1}$ representan el peso de interés del usuario antes de hacer una búsqueda y una vez realizada la búsqueda. Teniendo en cuenta que el tiempo empleado para el cálculo del peso en la búsqueda previa debe ser rápido, éste podría tener un pequeño impacto en la búsqueda actual dependiendo del comportamiento del usuario. Por lo tanto, el peso de interés del usuario actual debería ser multiplicado por un factor de atenuación.

Figura 5. Arquitectura para búsquedas personalizadas basadas en ontologías personalizadas



Se presentará cómo es integrado este servicio en una arquitectura definida para búsquedas personalizadas basadas en ontología personalizada.

La palabra clave es ingresada en primer lugar a la interfaz de consulta, luego es convertida en un conjunto de palabras claves después de que es procesada por el módulo de expansión de consultas. De acuerdo a la librería de ontología personalizada para cada una de las palabras claves en el conjunto es enviada a uno o más motores de búsqueda independientes. Todos los resultados son procesados en el módulo de síntesis de resultados de consulta. En el módulo de

organización de resultados de consulta, se ordenan los resultados por los pesos de interés del usuario según la librería de ontología personalizada.

3.1.4. Ontologías personalizadas para la personalización de búsquedas Web[71]

Este estudio se enfoca en el desarrollo automático de perfiles de usuario basado en ontologías personalizadas y ontologías de página. Se propone un nuevo sistema UCI (User Conceptual Index) que orienta la búsqueda al contexto y al individuo. UCI utiliza relaciones entre las consultas de búsqueda y las páginas visitadas por el usuario, así es como provee un ranking basado en el contexto individual de la búsqueda.

Obtención de los datos. Los datos se obtienen comúnmente de los logs del servidor (servidor) o del navegador del usuario (cliente).

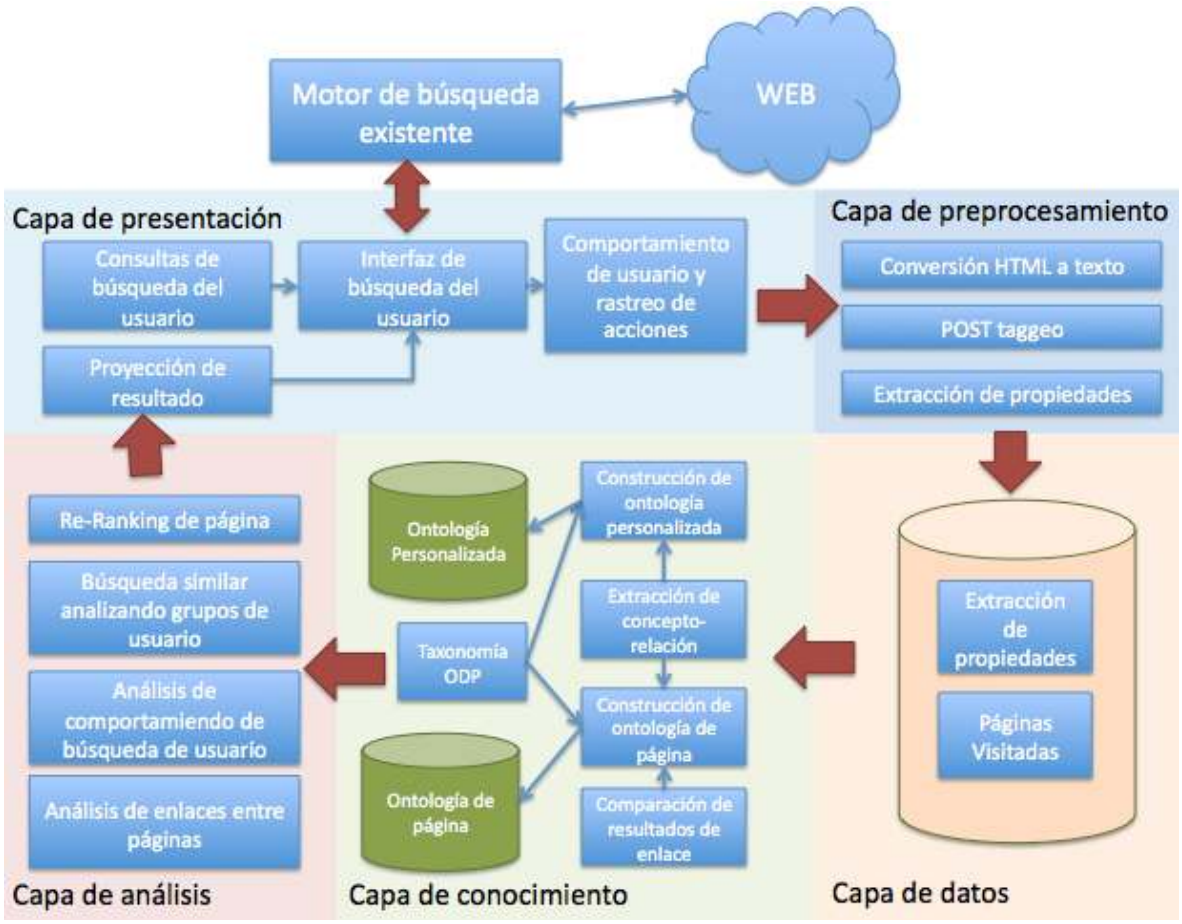
Servidor: Basado en[72] los datos no son confiables. Desafortunadamente los aciertos por caché faltan en estos logs y no hay exactitud en la cantidad de visitas por usuario. Propiedades temporales como tiempos de visita en las páginas son consideradas altamente informativas y deducen preferencias del usuario. Las Cockies que permiten rastreo inter-sesión de usuarios violan la privacidad del usuario. Las páginas que requieren registro por parte del usuario son comúnmente ignoradas. Las URLs dinámicas que tienen embebidas IDs de sesiones restringen cacheo intermedio y no manejan correctamente el intercambio de URLs entre las personas. Buscar en el caché también reduce el tiempo que se ha ganado que es precisamente para lo que el caché fue creado.

Teniendo en cuenta todos estos problemas del lado del servidor en esta propuesta se desea obtener una colección de datos del usuario por medio de su interacción

[71] Sendhilkumar, S. y Geetha, T. V. "Personalized Ontology for Web Search Personalization". Proceedings of the 1st Bangalore Annual Compute Conference, 2008.
[72] Shahabi, Cyrus y Banaei-Kashani, Farnoush. "A Framework for Efficient and Anonymous Web Usage Mining Based on Client-Side Tracking". Revised Papers from the Third International Workshop on Mining Web Log Data Across All Customers Touch Points, 2001, p. 113 - 144.

con el navegador Web. Se realizó una comparación evaluativa entre una búsqueda normal y una búsqueda basada en UCI y se notó que en conjunto los resultados de la búsqueda mejoraron un 9%. La inclusión de ontologías ayuda aún más a mejorar la búsqueda, para una mejor explicación del sistema veamos su arquitectura:

Figura 6. Arquitectura a 5 capas para la personalización de las búsquedas en la Web



La arquitectura propuesta se compone de 5 capas: Presentación, Preprocesamiento, Datos, Conocimiento y Análisis. Todas éstas se explicarán a continuación.

Capa de presentación. Es el lugar donde los usuarios ingresan sus consultas por medio de una interfaz de usuario diseñada para obtener la consulta y después de todo el proceso, retornar los resultados. Aquí se desarrolla la recolección de información del lado del cliente de tal modo que la interfaz es diseñada para mantener un rastreo total del usuario como páginas visitadas, tiempo invertido en la página, velocidad de scrolling y acciones del usuario como guardar, copiar, imprimir, agregar a favoritos, etc.

Capa de preprocesamiento. Obtiene el texto HTML, lo parsea removiendo las etiquetas HTML y lo convierte en un documento de texto. Despúes se hace un etiquetado POS (part-of-speech) que agrega una etiqueta nueva al documento de texto. Se extraen los nombres o sustantivos del texto y las fecuencias son computadas, las tres con más ocurrencias son seleccionadas como características/índices. Esto se repite para todas las páginas visitadas por el usuario.

Capa de datos. Ésta es la fuente de las dos capas posteriores de conocimiento y análisis. Cada búsqueda de usuario es manejada como una transacción que se define como un conjunto de páginas visitadas por el usuario y se representa así:

$$t = \{p_1, p_2, \ldots, p_n\} \qquad (7)$$

y cada sesión es identificada por un conjunto de transacciones que puede ser representada así:

$$s = \{t_1, t_2, \ldots, t_m\} \qquad (8)$$

Existirán entonces dos nuevas matrices Transacción-Característica TF y Sesión-Transacción ST. Ahora, después de combinar TF y ST se tiene la matriz Sesión-Característica SF $SF = \{s_1, s_2, \ldots, s_n\}$ donde cada s_i es un vector k-dimensional sobre el espacio de las características. Así cada sesión de usuario se puede

representar mediante un vector de características de contenido, reflejando los intereses del usuario a lo largo de la sesión.

El siguiente proceso es la computación del UCI. Para un conjunto dado de consultas de búsqueda SQ y un conjunto de palabras característica/índice IW, UCI se puede definir como la relación entre SQ y páginas relevantes, representada por la siguiente función:

$$UCI = f(W_{SKW}, IW) \quad (9)$$

Donde SKW es la consulta de búsqueda, IW las palabras índices para una página y W es la suma de los pesos de cada SQ y sus relevantes IW. El sistema identifica el par (SQ, IW) con el UCI más alto y lo recomienda primero al usuario. El UCI también toma en cuenta varios factores de personalización como son los aciertos, tiempos del usuario en la página, historia de navegación y usos de las consultas de búsqueda. La metodología propuesta al parecer funciona bien de acuerdo a experimentos realizados y mejora una búsqueda normal en un 9%, ver [55].

Capa de conocimiento. Aquí es donde entran las ontologías. Se construirán dos ontologías:

Ontología de página: es incremental y se construye de acuerdo a las páginas visitadas por el usuario.

Ontología personalizada: se construye de acuerdo a los datos del perfil de usuario que se encuentran en la capa de datos.

Veamos con más profundidad como se construyen:

Ontología de Página:

Figura 7. Ontología de página

Los conceptos extraídos durante el preprocesamiento son usados para construir la ontología de página. Para establecer las relaciones entre los conceptos se hace uso de un diccionario taxonómico y un diccionario convencional. Así se tienen los conceptos relacionados los cuales se le entregan a la herramienta Protégé [73] para construir la ontología. Las relaciones establecidas son de composición e identificación de hipónimos e hiperónimos.

El peso del enlace $W_{Pi \to Pj}$ entre dos páginas está dado por la ecuación:

$$\cos\theta(i,j) = \frac{\sum W_{mi} * W_{nj}}{\sqrt{\sum W_{mi}{}^2} * \sqrt{\sum W_{nj}{}^2}} \qquad (10)$$

Donde W_{mi} representa el peso del concepto m representando el i de la página P_i. Igual para W_{nj}. El peso de las palabras concepto/índice se obtiene así:

$$W = \frac{1}{n}\left[\sum_{j=1}^{n} F_j(IW_i) + \sum_{j=1}^{n} T_j(IW_i)\right] \quad (11)$$

Donde j indica el número de sesión, $F_j(IW_i)$ es la frecuencia de recuperación de la palabra índice i en la sesión j y $T_j(IW_i)$ es el tiempo de uso de la palabra índice i en la sesión j y n es el número total de sesiones de búsqueda.

Ahora UCI debe ser recalculado y mostrado así:

$$UCI = \Sigma(W_{SKW_i \rightarrow P_i}) \cdot \Sigma (W_{P_i \rightarrow P_j}) \quad (12)$$

El nuevo UCI tiene en cuenta los nuevos y antiguos intereses de usuario que pueden ser usados en el proceso de recomendación de tal manera que estas combinaciones se acepten o rechacen de acuerdo al peso.

Ontología personalizada:

Figura 8. Ontología personalizada

Se construye de los perfiles de usuario, cada nodo de la ontología personalizada se compone principalmente de los intereses de usuario y las páginas relevantes. A cada nodo se le asigna un peso personalizado que es computado de acuerdo a las acciones del usuario cuando está navegando. Las acciones consideradas y sus pesos son:

Guardar, altamente relevante con un peso de 1.

Imprimir, altamente relevante con un peso de 1.

Copiar, parcialmente relevante con un peso de 0.25 a 0.75 de acuerdo a la cantidad de contenido copiado.

Agregar a favoritos, relevante o parcialmente relevante con un peso de 1 si es usada en la misma sesión y 0.5 usada en otra sesión.

Capa de Análisis. Análisis de comportamiento son efectuados a un solo usuario o a grupos de usuarios, identificando patrones de consultas parecidos,

categorización de usuarios basada en comportamiento e intereses. Los resultados de este análisis son usados para la recomendación de páginas que son dadas por medio de la capa de presentación.

3.2. TÉCNICAS DE EXPANSIÓN DE CONSULTAS

La expansión de consultas es muy usada en los motores de búsqueda dado que las consultas cortas no poseen mucha información que permita retornar resultados satisfactorios. Las palabras claves no son siempre buenos descriptores de contenido y de lo que se desea buscar. También se da el caso en que los usuarios en muchas ocasiones usan las palabras incorrectas para hacer búsquedas.

Existen los tipos de consultas manuales y automáticas:

Consulta manual: El usuario tiene que intervenir y debe conocer el mecanismo del motor de búsqueda para desarrollar las consultas.

Consulta automática: Existen dos formas de expandir una consulta automáticamente, local y global.

En la *estrategia local* se tienen en cuenta logs, subconjuntos de documentos o retroalimentación de los usuarios antes de hacer las consultas. Entre sus desventajas está el exagerado consumo de tiempo y que los usuarios en la mayoría de los casos no estarán dispuestos a participar en la optimización de la selección de documentos o a dejar una opinión en estos.

Por su parte la *estrategia global* se basa en otras herramientas externas como los tesauros u onomásticos. Ésta representa un menor consumo de tiempo y mayor facilidad de implementación.

Entre los aspectos claves para la expansión de consultas están las fuentes de datos, los métodos que asignan peso a las relaciones y los tipos de integración que se usan para la expansión de términos.

Las estrategias automáticas para la expansión de consultas son comúnmente usadas en los motores de búsqueda, pero se identifica una carencia de relación semántica entre los términos y eso es precisamente lo que el uso de ontologías desea mejorar.

A continuación se mostrarán diferentes propuestas de expansión de consultas:

3.2.1. Expansión de Consultas mediante minería de logs[73]

La expansión de consultas basada en logs de usuarios ha mostrado muy buenos resultados, la propuesta consiste en un log de usuarios que guarda la interacción del usuario con el sistema de búsqueda.

Existen algunas retroalimentaciones que son introducidas por el usuario. Por ejemplo, la calificación (de 1 a 10) de la información y que tan útil ha sido, se hace un ranking y se obtiene una retroalimentación de pseudorelevancia (pseudoretroalimentación).

Generalmente se le da prioridad al ranking de acuerdo al número de clics del usuario en determinada búsqueda. Se asume que los documentos con un ranking más alto son los que han sido escogidos más veces por los usuarios y por consiguiente son más relevantes. El método anterior resulta muy efectivo, pero su desempeño depende de la calidad de la búsqueda inicial. Si la búsqueda inicial es vaga e incompleta perfectamente puede alterar todos los resultados de la consulta y por lo tanto el ranking.

La expansión de consultas por minería de logs tiene las siguientes propiedades:

[73] Hang Cui; Ji-Rong Wen; Jian-Yun Nie y Wei-Ying Ma. “Query Expansion by Mining User Logs”. IEEE Transactions on Knowledge and Data Engineering, Volume 15, Issue 4, 2003, p. 829-839.

- La correlación de términos encontrados puede ser pre-computada sin conexión, de esta manera el preproceso de pseudorelevancia no sería necesario.
- Los logs se crean de acuerdo a sesiones de diferentes usuarios. Así se espera por ejemplo que la palabra "Windows" en un grupo amplio de usuarios que buscaban el sistema operativo retorne al sistema operativo y no términos relacionados a la palabra como puerta, cortina, etc.
- Las relaciones pueden evolucionar dada la acumulación de logs, por lo tanto siempre habrá una actualización en los términos.

Como se ha expresado a lo largo del documento, una de las necesidades primordiales es el manejo de relaciones entre el espacio de la consulta y el espacio de los documentos almacenados. La idea de explotar logs de usuario precisamente apunta a ese objetivo, es decir, la creación de un puente entre los dos espacios. Para eso la primera tarea es extraer las *sesiones de consulta* de un conjunto grande y complejo de datos de logs. Las *sesiones de consulta* se definen de la siguiente manera:

$$sesion := <textoDeConsulta> [documentoAlCualSeLeHaDadoClic]^* \quad (13)$$

Cada sesión contiene una consulta y un conjunto de documentos a los cuales el usuario les ha dado clic. La idea central de este método es que, si un conjunto de documento es seleccionado frecuentemente por las mismas consultas, entonces los términos de los documentos están fuertemente relacionados con los términos usados en las consultas. Una importante suposición que se hace es que se considera a los documentos que se les ha dado clic relevantes para la consulta.

Estos son los pasos que se siguen en la expansión de una consulta basada en minería de logs:

1. Extraer los términos de la consulta que no aplican para esta misma.

2. Buscar todos los documentos relacionados con cualquiera de los términos de la consulta en las "sesiones de consulta".

Veamos como determinar los grados de correlación entre términos. Se definen estos grados como la probabilidad condicional entre términos. Por ejemplo, $P(w_j^{(d)}|w_i^{(q)})$ para cualquier término en un documento $w_j^{(d)}$ y cualquier término en una consulta $w_i^{(q)}$. La probabilidad $P(w_j^{(d)}|w_i^{(q)})$ puede ser determinada así (donde S es el conjunto de documentos que se les ha dado clic para las consultas que contienen el término de consulta $w_i^{(q)}$):

$$P(w_j^{(d)}|w_i^{(q)}) = \frac{p(w_j^{(d)}, w_i^{(q)})}{P(w_i^{(q)})}$$
$$= \frac{\sum_{\forall D_k \in S} P(w_j^{(d)}, w_i^{(q)}, D_k)}{P(w_i^{(q)})}$$
$$= \frac{\sum_{\forall D_k \in S} P(w_j^{(d)}|w_i^{(q)}, D_k) \times P(w_i^{(q)}, D_k)}{P(w_i^{(q)})} \quad (14)$$

Se asume que $P(w_j^{(d)}|w_i^{(q)}, D_k) = P(w_j^{(d)}|D_k)$. Esto significa que el documento D_k aparta al término de consulta del término del documento $w_j^{(d)}$. Por lo tanto:

$$P(w_j^{(d)}|w_i^{(q)}) = \frac{\sum_{\forall D_k \in S} P(w_j^{(d)}|D_k) \times P(D_k|w_i^{(q)}) \times P(w_i^{(q)})}{P(w_i^{(q)})}$$
$$= \sum_{\forall D_k \in S} P(w_j^{(d)}|D_k) \times P(D_k|w_i^{(q)}).$$

(15) $P(D_k|w_i^{(q)})$

Es la probabilidad condicional de que al documento D_k se le haya dado un clic cuando $w_i^{(q)}$ aparece en la búsqueda de la consulta del usuario.

$P(w_j^{(d)}|D_k)$ es la probabilidad condicional de ocurrencia de $w_j^{(d)}$ si el documento es seleccionado. $P(D_k|w_i^{(q)})$ y $P(w_j^{(d)}|D_k)$ pueden ser estimados de los logs de usuarios y de la frecuencia y ocurrencia de los documentos de la siguiente forma:

$$P(D_k|w_i^{(q)}) = \frac{f_{ik}^{(q)}(w_i^{(q)}, D_k)}{f^{(q)}(w_i^{(q)})} \quad (16)$$

$$P(w_j^{(d)}|D_k) = \frac{W_{jk}^{(d)}}{\sum_{\forall t \in D_k} W_{tk}^{(d)}} \quad (17)$$

donde:

$f_{ik}^{(q)}(w_i^{(q)}, D_k)$ es el número de sesiones de consulta en las cuales el término de consulta $w_i^{(q)}$ y el documento D_k aparecen juntos.

$f^{(q)}(w_i^{(q)})$ es el número de sesiones de consulta que contienen al término $w_i^{(q)}$.

$P(w_j^{(d)}|D_k)$ es el peso normalizado del término $w_j^{(d)}$ en el documento D_k, el cual es dividido por la suma de todos los pesos de los términos en el documento D_k.

La probabilidad de que un término de un documento sea seleccionado como un término de expansión dado en un término de una consulta se calcula con la siguiente ecuación:

$$P(w_j^{(d)}|w_i^{(q)}) = \sum_{\forall D_k \in S} \left(P(w_j^{(d)}|D_k) \times \frac{f_{ik}^{(q)}(w_i^{(q)}, D_k)}{f^{(q)}(w_i^{(q)})} \right)$$

(18)

la cuál es una combinación de las ecuaciones 15, 16 y 17.

3. Para cada término de estos documentos, usar

$$CoWeight_Q(w_j^{(d)}) = \ln\left(\Pi_{w_t^{(q)} \in Q}\left(P\left(w_j^{(d)}|w_t^{(q)}\right) + 1\right)\right)$$

(19)

para calcular su evidencia de que ha sido seleccionado como un término de expansión de acuerdo a la totalidad de la consulta.

4. Seleccionar los términos con el peso más alto de cohesión y formular una nueva consulta adicionando estos nuevos términos.

5. Retornar nuevos documentos de acuerdo a la consulta nueva.

3.2.2. Expansión de consultas con aprendizaje transductivo[74]

Las retroalimentaciones dadas por el usuario pueden llegar a ser muy importantes para la expansión de consultas, en este caso se busca encontrar los resultados más acertados con una mínima retroalimentación. A continuación se presenta el método paso a paso:

Ingreso de la consulta. El usuario empieza la transacción con la entrada de una consulta al sistema de recuperación de información (IRS Information Retrieval System).

Recolección de información. El usuario recibe la información retornada por el IRS y a su juicio escoge los documentos más relevantes en orden descendente. Se espera que el usuario suspenda la búsqueda de documentos cuando encuentre alguno suficientemente relevante. Basado en estos documentos: el relevante y los menos relevantes el sistema ahora cuenta con los documentos mínimos para usar aprendizaje transductivo.

Método de cálculo de documentos más relevantes. Un algoritmo de aprendizaje transductivo es usado para la búsqueda de documentos relevantes que el usuario no encontró. Se definen los documentos "relevantes" e "irrelevantes" de acuerdo a una entrada anterior de documentos. Teniendo en cuenta sólo los documentos relevantes el IRS calcula un resultado w_t para cada término de estos documentos. Los w_t más altos son seleccionados para la expansión.

Estos nuevos términos son usados para la expansión de la consulta y son entregados al IRS para obtener una nueva búsqueda. La escogencia de

[74] Okabe, Masayuki y Yamada, Seiji. “Semisupervised Query Expansion with Minimal Feedback”. IEEE Transactions on Knowledge and Data Engineering, Volume 19, Issue 11, 2007, p. 1585-1589.

los m más altos se basa en la función propuesta por Robertson en el método WPQ:

$$score(t) = \left(\frac{r_t}{R} - \frac{n_t - r_t}{N - R}\right) * \log \frac{r_t/(R - r_t)}{(n_t - r_t)/(N - n_t - R + r_t)} \quad (20)$$

donde r_t es el número de documentos relevantes que contienen al término t, n_t es el número de documentos que contienen al término t , R es el número de documentos relevantes para la consulta y N es el número de todos los documentos de la colección.

Método de aprendizaje transductivo. Es una máquina de aprendizaje técnica basada en la transducción que crea la clasificación de etiquetas para un conjunto de datos de prueba directamente sin hacer ninguna función aproximada de un conjunto de datos. La tarea de aprendizaje se define en un conjunto de datos X de n puntos. X consiste en un conjunto de datos $L = \{x_1, x_2, ..., x_l\}$ y un conjunto de datos de prueba $U = \{x_{l+1}, x_{l+2}, ..., x_{l+u}\}$, donde $l << u$. El propósito del aprendizaje es asignar una etiqueta a cada dato en U bajo la condición de que la etiqueta de cada dato esta contenida en L.

El aprendizaje inductivo consiste en dos fases: una fase de aprendizaje y una fase de inferencia. En la fase de aprendizaje, un subconjunto de datos L y etiquetas Y_L son dados como ejemplos. El que aprende producirá un modelo para predecir etiquetas por el resto de los puntos en U. Usando el modelo de inferencia se puede finalmente obtener las etiquetas para Y_U. En contraste con el aprendizaje inductivo, el transductivo usa L y U.

El aprendizaje transductivo no distingue entre las dos fases que usa el inductivo, esto pasa al mismo tiempo en una configuración transductiva. El aprendizaje transductivo aplica muy bien para cuando $|L|$ es muy pequeño. Para aplicar

aprendizaje transductivo en este caso se escoge SGT, un estado del arte en un algoritmo de aprendizaje transductivo. SGT[75] formaliza el problema de asignar etiquetes a U.

El algoritmo SGT se sale del alcance de este documento debido a que en ninguna parte hace uso de ontologías, para un acercamiento más amplio referirse a [59]. En parte, este método ha sido propuesto en este documento para mostrar una forma diferente de expansión de consultas, pero su explicación a fondo se sale del objetivo que es la recopilación de técnicas de recuperación de información basadas en ontologías. Sigamos entonces con el desarrollo normal del documento después de haber propuesto un método muy interesante del que se pueden basar nuevas ideas aplicadas a las ontologías.

3.2.3. Expansión de consultas basada en ontologías en un motor de búsqueda vertical[76]

Después de haber analizado métodos de expansión de consultas que presentan acercamientos muy interesantes, es hora de mostrar un método que sea capaz de hacer lo mismo basándose en "conocimiento" para poder hacer búsquedas más específicas y precisas. Precisamente eso es lo que hace el siguiente método, usar conocimiento que se encuentra formalizado en ontologías.

Cuando se tiene una consulta por lo general esta carece de información adecuada para saber lo que se desea buscar, por esta razón antes de hacer la consulta este método la extiende de manera que sus resultados sean más específicos.

Describamos de manera general este método:

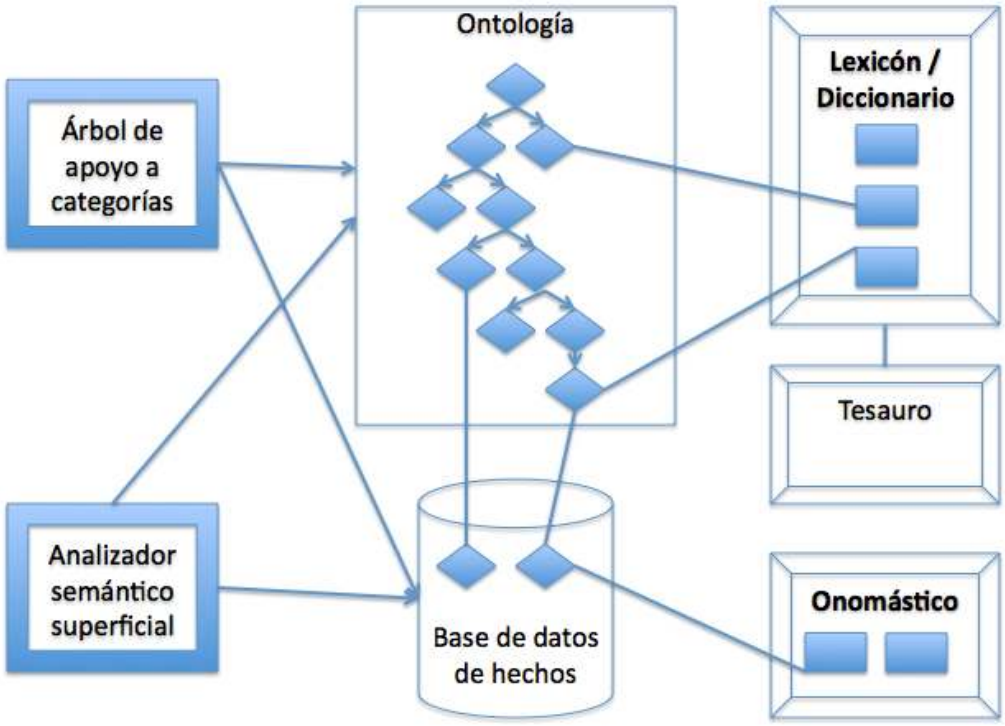
- Hace uso de una base de datos de conocimiento (Domain knowledge data base).

[75] Joachims, T. “Transductive Learning via Spectral Graph Partitioning”. Proceedings International Conference Machine Learning. 2003. p. 143-151.
[76] Liangjun Ma; Lin Chen; Yibo Gao y Yiping Yang. “Ontology based Query Expansion in Vertical Search Engine”. Proceedings of the Sixth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 02. 2009, p. 285-289.

- Análisis sintáctico (parsing) de las consultas en términos de construcción semántica de un grafo, esto se hace utilizando la base de datos de conocimiento.
- Cálculo de distancia semántica.
- Cambio de términos del grafo semántico con operadores lógicos y obtención de resultado.

Base de datos de conocimiento. Se tienen los módulos de ontología, lexicón, tesauro, onomástico, árbol de apoyo a categorías, base de datos de hechos y un analizador semántico superficial.

Figura 9. Base de datos de conocimiento



El analizador semántico superficial se encarga de *parsear* una consulta. Por ejemplo, "1000 Dollars, computer " que es la consulta del usuario para el sistema se convierte en "price:1000 Dollars", "item:computer".

La ontología puede ser dividida en dos clases, de acuerdo con los hechos que describe, una es la ontología principal que es el estándar básico de la mayoría de material clasificado y la otra es la ontología de apoyo en la cual los elementos son candidatos para valores de instancias de la ontología principal. Para estos valores que no pueden ser descritos como ontologías pero que pueden expresar algún tipo de jerarquía existe el árbol de apoyo a categorías.

Grafo semántico. El grafo semántico se puede definir así:

$$SD:=(V,E) \tag{21}$$

Donde:

V es el vértice del grafo que puede representar una palabra, frase o cadena.

E son las aristas que se pueden definir así:

$$Eij=\{v_i,v_j,r_{ij},w_r\} \tag{22}$$

donde
v_i y v_j son los respectivos vértices de la relación que son conectados por la arista Eij; e i es la distancia del vértice v_i a el vértice inicial v_0. Y la distancia de v_i y v_0 es siempre garantizada cómo la ruta más corta entre los dos vértices.

r_{ij} es la relación entre los dos vértices y puede ser de varios tipos:

- hiperonimia

- homonimia
- similitud
- atributos, aquí caben los conceptos de clase, instancia y valor del atributo.

w_r es el peso asignado a la relación r entre v_i y v_j.

Distancia semántica. Se define como distancia semántica la distancia entre el vértice inicial v_0 y cualquier otro vértice v_i:

$$S_i = -log_2\{\prod_{n=1}^{i} T_n(\frac{k}{i+\lambda})^m\} + 1 \\ = -\sum_{n=1}^{i} log_2 T_n - mlog_2(\frac{k}{i+\lambda}) + 1 \quad (23)$$

donde T_n es el peso de la relación entre los vértices v_{n-1} y v_n en la ruta más cercana de v_0 a v_i, la cuál se encuentra entre [0,1], especialmente cuando $n-0$, $T_n=1$. k y λ son enteros diferentes. m es un factor de atenuación y es un entero y $m>2$. Para cada uno de los dos vértices en el grafo suponiendo v_i y v_j e $i>j$ la distancia semántica es:

$$S_{ij} = S_i - S_j = \sum_{n=1}^{j} log_2 T_n + mlog_2 \frac{k}{i+\lambda} \\ - \sum_{m=1}^{i} log_2 T_n - mlog_2 \frac{k}{j+\lambda} \quad (24)$$

donde T_n es el peso de la relación entre los vértices v_{n-1} y v_n en la ruta más cercana de v_0 a v_i.

Descripción del algoritmo. Asumiendo a Q como la consulta inicial

1. Se pasa Q al analizador semántico superficial y este a su vez entrega un conjunto A de pares de atributos $A=\{a:a_1,b:b_2,...,n:m_n\}$. Los elementos que cacen deben ser eliminados de Q.

2. Basados en el onomástico se obtienen las entidades N de lo que queda de Q, se borran los elementos que coinciden.

3. Sea $Q^{'}$ lo que queda, $Q^{'} = \{w_1, w_2, ..., w_n\}$ es la nueva consulta inicial y w_i es la i de todas las posibles combinaciones de palabras en $Q^{'}$.

4. Para cada pareja de atributos en A, y cada elemento en $Q^{'}$ se construye un grafo semántico. Se obtiene entonces una colección de grafos semánticos:

$$SD = \{sd | inivertex(sd) = i, i \in Aori \in Q'\} \tag{25}$$

donde

$inivertex(sd) = i$ significa que i es el vértice inicial del grafo semántico sd.

Para generar el grafo semántico se requiere seguir la siguiente regla:

Regla 1: Para generar los sub-nodos de v_i en un grafo semántico, si v_i es un nodo en la ontología de apoyo o en el árbol de apoyo a categorías, todos los nodos en la ontología que tengan valores de atributos iguales a v_i o a sus sub-nodos deberían ser agregados al grafo semántico como sub-vértices de v_i.

5. Calcular la distancia semántica entre cualquier vértice y el vértice inicial en cada grafo semántico de acuerdo a las ecuaciones 23 y 24, compararla con el umbral anterior. Seleccionar los vértices para los cuales la distancia semántica sea menor que las del umbral y así ir reemplazando hasta encontrar la más óptima.

Para cada grafo semántico se obtiene:

$$EP_i = \{v_j | S_j < S', v_j \in sd_i\} \tag{26}$$

donde EP_i son los candidatos obtenidos del grafo semántico sd_i. S_j es la distancia entre el vértice v_j y el vértice inicial de sd_i. S' es el umbral que aún queda por recorrer.

6. Se obtienen los resultados temporales para obtener EPQ.

$$EPQ = \bigcap EP_i \tag{27}$$

Se deben tener en cuenta las siguientes reglas en el momento de hacer la gran intersección:

Regla 2: Cuando $v_i \cap v_j$ y v_i y v_j están ambos nodos en la ontología principal o en la ontología de soporte entonces:

$v_i \cap v_j = v_i$ si v_i es sub-nodo de v_j;

$v_i \cap v_j = v_j$ si v_j es sub-nodo de v_i;

$v_i \cup v_j = \phi$ si v_i no es sub-nodo ó super-nodo de v_j;

Regla 3: Cuando $v_i \cup v_j$ y v_i y v_j están ambos nodos en la ontología principal o en la ontología de soporte entonces:

$v_i \cup v_j = v_j$ si v_i es sub-nodo de v_j;

$v_i \cup v_j = v_j$ si v_j es sub-nodo de v_i;

Regla 4: Cuando $v_i \cap v_j$ y v_i y v_j están ambos nodos en la ontología principal o en la ontología de soporte entonces:

Si v_i es igual a v_j , $v_i \cap v_j - v_i - v_j$; sino es ϕ

7. Se entregan los últimos resultados de esta gran intersección y se pasan al motor de búsqueda para que muestre un resultado.

3.2.4. Modelo de Expansión de Consultas basado en Ontología[77]

El modelo que se describirá a continuación logra la extracción de una consulta semántica y una expansión a nivel semántico de los conceptos a través del emparejamiento de los conceptos consultados y el dominio ontológico que tienen. El resultado de esta expansión es un conjunto de conceptos los cuales son extraídos de la librería de ontologías donde las similitudes semánticas de estos son mayores que un umbral definido.

Para entender mejor el modelo de expansión de consultas basado en ontologías vamos a tener las siguientes definiciones:

Asumamos que α es el umbral de similitud y que β es la precisión mínima de α. El número de conceptos de la similitud semántica de los cuales son mayores que el umbral es n. Entonces la densidad de un concepto es:

$$\rho = \frac{N}{1-\alpha} \qquad (28)$$

Si $\rho \geq \frac{N}{\beta}$, entonces debe haber más de N conceptos en un rango de β entre 1 y α.

La función de dispersión usando la distancia semántica como su parámetro cambia esencialmente la distribución de las similitudes volviendo a calcular la similitud semántica de los conceptos. Esto también le ayuda al umbral de similitud

[77] Zhijun Zhang y Yunqing Li. "Research and Implementation of the Personalized Meta Search Engine based on Ontology", Web Intelligence and Agent Systems, Volume 1 , Issue 3-4, 2003, p. 219 - 234.

a controlar el tamaño de los conceptos que estén bien establecidos y mejora la eficiencia de los servicios de recuperación de información. Por lo tanto se deben satisfacer las siguientes características:

- Cuando la distancia semántica es 0, la similitud es 1. Es decir, cuando los dos conceptos son el mismo, la similitud es 1.
- La similitud decrece cuando la distancia semántica incrementa. Es decir, entre más grande es la distancia de dos conceptos más pequeña es la similitud entre estos.
- La salida de la función debe estar en el rango [0,1].

Se han escogido 3 funciones de dispersión diferentes:

$$Sim_1 = \frac{1}{Dist(A,B)+1} \quad (29)$$

$$Sim_2 = \frac{1}{Dist^2(A,B)+1} \quad (30)$$

$$Sim_3 = \frac{1}{e^{Dist(A,B)+1}} \quad (31)$$

Se verifica que las tres funciones cumplen las características mencionadas previamente. La diferencia de las funciones está en la razón con la que incrementa el valor con la disminución de la distancia semántica. La primera es una función lineal decreciente, la segunda decrece más rápido que la primera y la última es la que más rápido decrece.

Cuando todos los elementos del conjunto de conceptos de expansión han sido generados, para cada uno de los elementos del conjunto calculamos la densidad de los conceptos y los comparamos con ρ_0 el cual es el umbral de la densidad. Si

la densidad el concepto es mayor que ρ_0, la similitud semántica se volverá a calcular por las funciones de dispersión. Diferentes tipos de conjuntos pueden usar la función de dispersión independientemente.

Una estrategia de búsqueda basada en las librerías ontológicas puede ser introducida después de la combinación de los dos aspectos discutidos. Primero averiguamos el concepto que es sinónimo de las palabras originales en la librería de ontología y los agregamos en la colección A. Segundo, seguimos el siguiente algoritmo que es implementado por cada concepto C en A.

Entrada: concepto C, el umbral α y β, el umbral ρ_0 de la
densidad
Salida: el conjunto del concepto de expansión GP
Estados iniciales: el conjunto GP y T_i están vacíos,
la similitud semántica S es 0.

PARA cada concepto en A
COMIENZO
PARA i=1 **A** n
COMIENZO
PARA todos los conceptos con la distancia más corta siendo i
COMIENZO
Calcular la similitud semántica S
SI $S \geq \alpha$ **ENTONCES**
Agregar el concepto a T_i de acuerdo a los tipos
DE OTRO MODO
Dejar el concepto
FIN
SI T_i no tiene cambios **ENTONCES**
Salir del ciclo actual

FIN

FIN

PARA j=1 **A** n

COMIENZO

Ordenar los conceptos en T_j de forma ascendente

MIENTRAS (T_j no esté vacío)

COMIENZO

Mover el primer concepto desde T_j al conjunto GP

Calcular la densidad del concepto ρ

SI $\rho \geq \rho_0$ **ENTONCES**

COMIENZO

Calcular de nuevo la similitud(P) del primer concepto con la función de dispersión

SI $P \geq \alpha$ **ENTONCES**

Mover el primer concepto desde T_j al conjunto GP

DE OTRO MODO

Salir del ciclo actual

FIN

FIN

FIN

Para ver cómo se integra este algoritmo con una arquitectura basada en ontologías personalizadas ver la figura 4.

3.2.5. Expansión de consulta personalizada basada en ontologías[78]

Varias ideas se han expuesto hasta el momento, la mayoría basadas en información que se obtiene por la navegación del usuario en el navegador Web.

[78] Calegari, Silvia y Pasi, Gabriella. "Personalized Ontology-based Query Expansion". Web Intelligence & Intelligent Agent, Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03, 2008, p. 256-259.

Esta propuesta muestra un acercamiento diferente a la recuperación de información basada en ontologías. Consiste en el desarrollo de un "gadget" para Google llamado Contexto de Información Personal (PIC, Personal Information Context) el cual se usa para expandir las consultas del usuario semánticamente.

El objetivo es definir una ontología personalizada con las consultas realizadas anteriormente por el usuario. Para obtener esto se consideran algunas propiedades de O-FCN (Object-Fuzzy Concept Network) que se basa en relaciones "borrosas" no taxonómicas llamadas correlaciones.

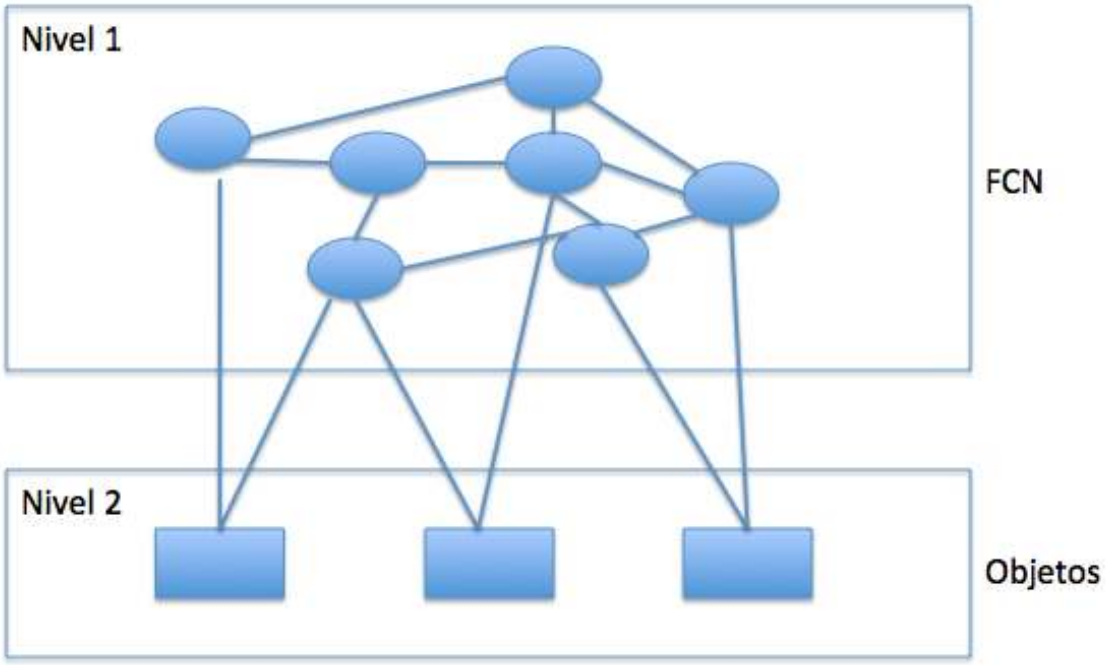
La idea es ayudar al usuario durante la definición de su consulta de una manera transparente. El usuario directamente interactúa con O-FCN para formular la consulta y la expansión de la consulta es llevada a cabo no solo con el análisis de correlaciones sino también con conceptos almacenados en el computador del usuario.

Descripción de O-FCN. Para este caso se usa O-FCN, una ontología "borrosa" liviana. Los conceptos son enlazados con por medio de una correlación.

$$corr: C \times C \rightarrow [0,1] \quad (32)$$

Donde $C = \{c_1, c_2, \ldots, c_n\}$ es el número de conceptos contenidos en la ontología "borrosa". Lo más cerca a 1 es el valor de $corr$, mientras éste sea más alto significa que más valores están asociados. Se quiere generar O-FCN en el cual una correlación sea definida con las consultas previamente realizadas por el usuario. Veamos básicamente como funciona un O-FCN:

Figura 10. O-FCN



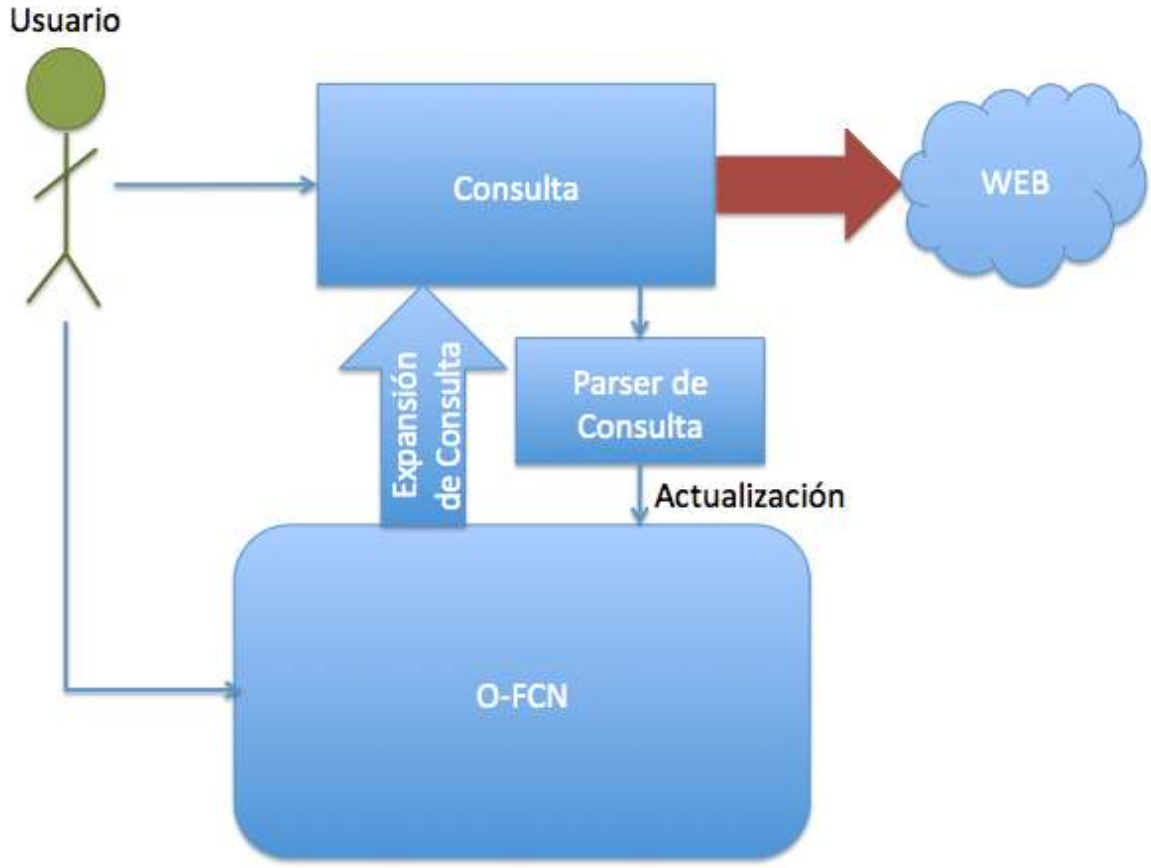
Se tiene que O-FCN es un grafo pesado $\vartheta_{FG} = \{O_{DB}, N_F\}$ donde O_{DB} es un conjunto de objetos almacenado en un repositorio y $N_F = \{C, F, F_W\}$ es una red de conceptos "borrosos"(FCN). Cada objecto es decrito por los conceptos de FCN, así por ejemplo : $\forall o_i \subset O_{DB} o_i - \{c_1, \ldots, c_n\}$ donde $c_1, \ldots, c_n \in C$.

O_{DB} es el conjunto de documentos almacenados en el repositorio, F identifica las aristas de FCN, define las relaciones de correlación entre los conceptos en C. Por ejemplo, $F := corr$. F_W identifica el caso especial de las relaciones de correlación en donde solamente los conceptos han sido escritos en la consulta.

Se definen entonces dos capas de conocimiento: una para los conceptos y otra para los objetos. Las aristas con peso igual a 0 son omitidas. El valor de F_W identifica la fuerza de las correlaciones de los conceptos y el diferente tamaño de los conceptos.

Uso de PIC para la expansión de las consultas. Como se había dicho PIC es un "gadget" para Google Desktop que busca con el uso de O-FCN facilitar las búsquedas del usuario y hacerlas más dinámicas teniendo en cuenta que el usuario cada día aporta nuevas consultas a la red. También se tienen en cuenta los documentos que están almacenados en el computador del usuario y que se obtienen por medio de Google Desktop API. Esta es su arquitectura:

Figura 11. Arquitectura de PIC (Personal Information Context)



El usuario se puede relacionar de dos formas con el sistema. La primera es ingresando la consulta de acuerdo a su experiencia y la segunda es teniendo la ayuda de O-FCN para identificar conceptos. El analizador sintáctico de consultas

se encarga de analizar la consulta y actualizar la correlación entre los conceptos. Después se tiene la expansión de la consulta que es la que se entrega a Google.

FCN en PIC funciona de tal forma que si el usuario ingresa las palabras caballo, paloma, la correlación va a ser $corr(caballo,paloma)=1$, ahora se tiene una nueva consulta blanco y paz $corr(blanco,paz)=1$. La semántica del grafo se va a enriquecer semánticamente así con una nueva consulta paloma y blanco por lo que se tendría $corr(paloma,blanco)=1$. Ahora todo el grafo se encuentra correlacionado.

Así se tiene como el usuario ha podido de alguna manera mejorar sus experiencias de búsqueda de acuerdo a sus preferencias, ayuda para completar consultas con O-FCN y también la gran ventaja de tener mapeados todos los documentos del sistema por medio de Google Desktop, lo que permite realizar una búsqueda más personalizada.

3.3. TÉCNICAS DE INDEXACIÓN Y RANKING

Las técnicas de indexación y ranking se han utilizado por los diferentes motores de búsqueda tradicionales y basados en ontología como parte esencial en la recuperación de la información de manera eficiente. Vamos a ver el caso de Swoogle con su algoritmo de ranking sobre los documentos semánticos encontrados en la Web.

Estas técnicas permiten priorizar las búsquedas que hace el usuario de manera que los primeros resultados sean efectivamente lo que el usuario quiere encontrar.

3.3.1. Motor de búsqueda de ontologías (Swoogle)[79]

Swoogle es un motor de búsqueda de documentos de Web semántica (SWDs, Semantic Web documents), que utiliza un sistema de recuperación y rastreo basado en indexación para encontrar dichos documentos en la Web, indexar sus metadatos y responde preguntas sobre éstos.

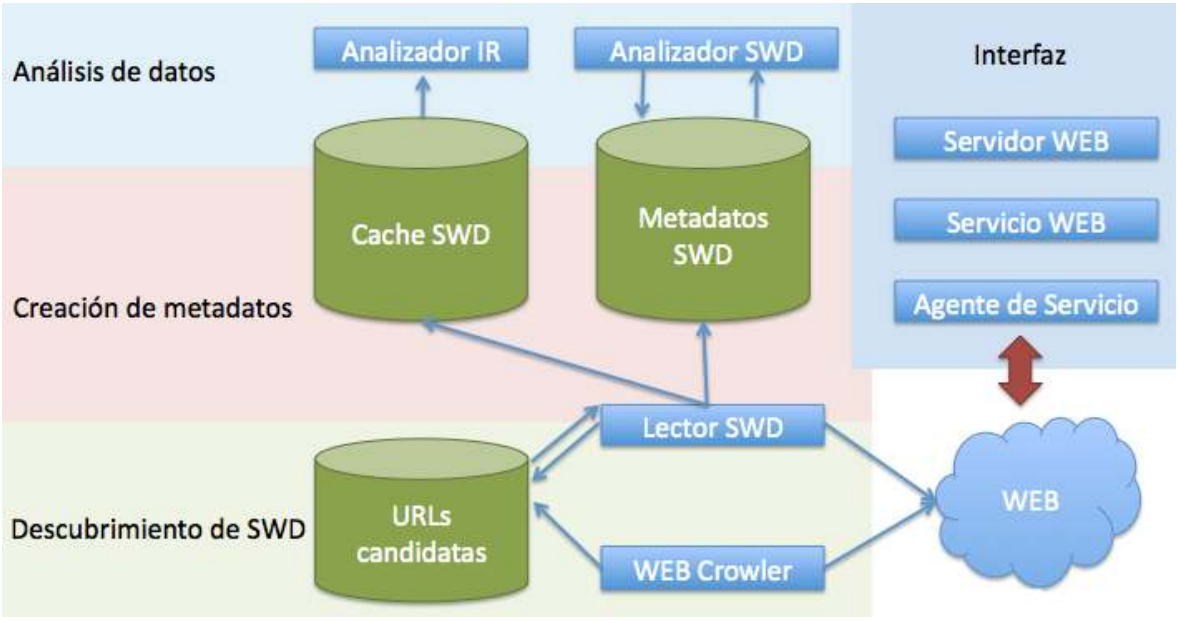
Swoogle diferencia dentro de su motor tres tipos de documentos que describiremos a continuación:

- Los documentos de Web Semántica (Semantic Web Document) son desarrollados en un lenguaje de semántico (rdf, owl, etc) que se encuentran en línea y son accesibles por usuarios de la Web y agentes de software.
- Ontologías de Web Semántica (Semantic Web Ontologies) son documentos con una proporción significativa en las declaraciones que define nuevos términos y extiende la definición de términos ya definidos en otros SWD adicionando nuevas propiedades o restricciones.
- Por último tenemos las base de datos de Web semántica (Semantic Web Databases) que se pueden considerar en esta clasificación como aquellos documentos que no definen o extienden un número significativo de términos. Una SWDB puede introducir individuos y hacer afirmaciones acerca de individuos definidos en otros SWD.

A continuación se presenta la arquitectura utilizada de Swoogle:

[79] Li Ding; Tim Finin; Anupam Joshi; Rong Pan; R. Scott Cost; Yun Peng; Pavan Reddivari; Vishal Doshi y Joel Sachs. “Swoogle: a search and metadata engine for the semantic web”, Proceedings of the thirteenth ACM international conference on Information and knowledge management, 2004.

Figura 12. Arquitectura utilizada por Swoogle



Swoogle tiene una técnica de ranking de los documentos semánticos que encuentra en la Web que le ayuda a determinar la importancia que tienen los documentos para priorizar la información que se le entrega al usuario. A continuación se presenta como funciona esta técnica:

Dado un SWD A y un SWD B, Swoogle clasifica los enlaces entre los SWD en cuatro categorías:

$imports(A,B)$, donde A importa todo el contenido de B.

$uses-term(A,B)$, donde A usa alguno de los términos definidos por B.

$extends(A,B)$, donde A extiende las definiciones de los términos definidos en B.

$asserts(A,B)$, donde A hace afirmaciones acerca de los individuos definidos por B.

Estas relaciones podrían ser tratadas de otra manera. Por ejemplo, el usuario que navega por la Web y observa $imports(A,B)$ mientras visita A, es natural para él seguir esta relación porque B es semánticamente parte de A. Del mismo modo, el usuario que navega por la Web puede seguir la relación $extends(A,B)$ porque éste puede entender el término definido completamente solo cuando navega en A y B. Por lo tanto, se asignan diferentes pesos para las cuatro categorías de relaciones entre los SWD.

Entre más términos haya en B referenciados por A, es más probable que el usuario siga el enlace de A a B. Basados en las consideraciones recién mencionadas, dado un SWD a, Swoogle computa su ranking usando la siguiente ecuación:

$$
\begin{aligned}
rawPR(a) &= (1-d) + d \sum_{x \in L(a)} rawPR(x) \frac{f(x,a)}{f(x)} \\
f(x,a) &= \sum_{l \in links(x,a)} weight(l) \\
f(x) &= \sum_{a \in T(x)} f(x,a)
\end{aligned}
\qquad (33)
$$

Donde $L(a)$ es el conjunto de SWDs que se relacionan con a, y $T(x)$ es el conjunto de SWDs con los que x se relaciona.

3.3.2. Modelo de recuperación de información basado en ontología para la Web semántica[80]

La recuperación de información de la Web todavía es una necesidad insatisfecha. Por ejemplo, si se quiere encontrar documentos de Beijing la capital de China , en la Web se puede utilizar "Beijing" como palabra clave. La recuperación en la Web por los motores de búsqueda daría como respuesta muchos documentos irrelevantes como "Universidad de Beijing", "Hoteles en Beijing". Se puede ver que

[80] Song Jun-feng; Zhang Wei-ming; Xiao Wei-dong; Li Guo-hui y Xu Zhen-ning. “Ontology-Based Information Retrieval Model for the Semantic Web”. IEEE International Conference on e-Technology, e-Commerce and e-Service, 2005.

la precisión en el proceso de recuperación de información todavía es baja. La razón principal por la que esta situación ocurre, es porque los sistemas actualmente no pueden entender la semántica en los contenidos Web.

Para entender el modelo de recuperación de información basado en ontologías para la web semántica que se mostrará a continuación, se debe tener en cuenta las siguientes definiciones:

- El dominio es una sección del mundo sobre la que se quiere expresar algún conocimiento. La conceptualización de dominio es abstraer un conjunto de términos y un conjunto de conocimientos desde el dominio en términos de las tareas para ser resuelto. Ontología de dominio es el conjunto de términos de dominio y el conjunto de conocimientos de dominio, esto es especificar explícitamente la conceptualización del dominio. Usualmente, se usa el lenguaje ontológico para anotar esta especificación. Ontología es una especificación explicita de la concepción del mundo. En la práctica, la ontología de dominio o la integración de varias ontologías de dominio son requeridas.
- Supongamos que el dominio D puede ser dividido en n subdominios, entonces la ontología de dominio de D se puede obtener integrando las ontologías de dominio de esos n subdominios.
- Después de que la ontología es generada, los términos en la ontología son usados como metadatos para marcar el contenido de la Web. Estas marcas semánticas son términos de indexación semánticos para la recuperación de información basada en ontología. Los términos de indexación semánticos son identificados a través de URIs.

Supongamos que X_i es un término de indexación semántico, la clase equivalente de X_i es $[X_i]$, $X_{i1} \in [X_i]$, $X_{i2} \in [X_i]$. Supongamos que en un

documento d_j, x, y y z denotan 3 cadenas de caracteres diferentes presentes en d_j, las marcas semánticas para x, y y z son X_{j1}, X_{j2}, X_{j3} respectivamente. Aunque x, y y z tienen diferentes apariencias desde el punto de vista de la semántica x, y y z son iguales, estos pueden ser representados por el mismo término de indexación semántico X_j. Supongamos una colección de documentos en la Web semántica es D, $D = \{d_1, d_2, \ldots, d_n\}$, $d_j \in D (1 \ldots j \ldots n)$.

Ahora supongamos que todas las clases equivalentes de los términos de indexación semántica para la colección de documentos D pueden ser representadas como un conjunto: $\{[X_1], [X_2], \ldots, [X_t]\}$, la frecuencia total de todos los elementos desde la clase equivalente $[X_i] (1 \ldots i \ldots t)$ presente en todas las marcas semánticas del documento d_j es f_{ij} (por ejemplo, el número total de veces que todos los elementos desde la clase equivalente $[X_i]$ están presentes en todas las marcas semánticas del documento d_j), como se muestra en la siguiente tabla:

Tabla 2. Frecuencia total de todos los elementos desde la clase equivalente presente en todas las marcas semánticas de un documento.

Clase Equivalente	d_1	d_2	...	d_j	...	d_n
$[X_1]$	f_{11}	f_{12}	...	f_{1j}	...	f_{1n}
$[X_2]$	f_{21}	f_{22}	...	f_{2j}	...	f_{2n}
...	...	...	...	...	...	...
$[X_i]$	f_{i1}	f_{i2}	...	f_{ij}	...	f_{in}
...	...	...	...	...	...	...
$[X_t]$	f_{t1}	f_{t2}	...	f_{tj}	...	f_{tn}

Tomando el esquema tf-idf introducido en[81]. Se tienen las siguientes fórmulas (n_i es el número de documentos en los cuales los elementos de la clase equivalente $[X_i]$ están presentes):

$$\begin{cases} tf_{ij} = \dfrac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{tj}\}} \\ idf_i = \log_2\left(\dfrac{n}{n_i}\right) \\ \omega_{ij} = tf_{ij} \times idf_i \end{cases} \qquad (34)$$

Utilizando estas fórmulas se pueden calcular todos los pesos, así la vista lógica de cada elemento en D se puede obtener en la siguiente ecuación:

$$d_j \cong (\omega_{1j}, \omega_{2j}, \ldots, \omega_{tj}) \qquad (35)$$

Esta vista lógica refleja la semántica de los términos de indexación semántica, de esta forma se representa adecuadamente un documento.

Para la información de usuario se necesita q, se puede obtener una vista lógica de q de acuerdo al conjunto de las clases equivalentes de los términos de indexación semántica: $\{[X_1], [X_2], \ldots, [X_r]\}$:

$$q \cong (\omega_{1q}, \omega_{2q}, \ldots, \omega_{tq}) \qquad (36)$$

Esta vista lógica refleja la semántica de q, de esta forma se representa adecuadamente la necesidad de la información de usuario.

La función de ranking puede ser:

[81] Baeza-Yates, Ricardo y Ribeiro-Neto, Berthier. Modern Information Retrieval, Addison Wesley, 1999.

$$\frac{1}{\sqrt{\sum_{i=1}^{t}(\omega_{iq}-\omega_{ij})^2}} \qquad (37)$$

La similitud entre q y d_j puede ser calculada de acuerdo a la función de ranking. Esta función de ranking define un ordenamiento entre los documentos en D con respecto a q. Como las vistas lógicas de documentos e información de usuario puede representar documentos y la necesidad de información de usuario de forma adecuada, entonces el ordenamiento es preferible y necesario.

A continuación se presentan partes fundamentales del modelo de recuperación de información basado en ontologías.

Figura 13. Arquitectura para un modelo de recuperación de información basado en Ontología

3.4. OTRAS TÉCNICAS QUE UTILIZAN ONTOLOGÍAS

En las secciones anteriores se han mostrado técnicas que utilizan la expansión de consultas basadas en ontologías, los perfiles de usuario, las ontologías personalizadas, ranking e indexación. En esta sección vamos a mostrar las técnicas de recuperación de información que aprovechan de manera eficiente las arquitecturas basadas en ontologías.

Las ontologías permiten la creación de algoritmos que facilitan la recuperación de información en motores de búsqueda basados en ontologías. Una ontología construida bajo una estructura de datos en particular sería aprovechada de forma eficiente por las técnicas de recuperación de información dado que conocen como está definida.

3.4.1. Algoritmo para la construcción de ontologías basada en Hipercubos[82]

El algoritmo que se describe a continuación crea un hipercubo con las palabras claves (KEYWORDs) que están presentes en la ontología. Vacío (EMPTY) especifica que no hay una palabra clave en la ontología. Activo (ACTIVE) significa que la palabra clave es mucho más activa y se puede trabajar con ella. Nuevo (NEW) significa que la palabra clave acaba de llegar y quiere ser asociada al hipercubo. Enlace (LINK) denota el enlace que hay entre las palabras en la ontología ya sea por atributos o propiedades similares. Modificar (MODIFY) significa que la estructura debe ser integrada a una dimensión superior desde la dimensión actual. Copiar (COPY) denota la réplica de una palabra clave ya existente la cual es insertada en la estructura para hacerla más estable y tener los enlaces bien definidos.

Entrada: Palabra clave (KEYBOARD)

Salida: Palabras claves enlazadas ó Estructura de hipercubo

[82] Mukhopadhyay, Debajyoti; Chakravorty, Sounak y Nandy, Sudarshan. “An Algorithm for Hyper-cube-based Ontology Construction”. Proceedings of the 10th International Conference on Information Technology, 2007, p. 283-288.

COMIENZO

Paso 1,2:

SI (KEYWORD=0) **ENTONCES**

RETORNA EMPTY

Paso 3,4:

SI (KEYWORD=1 **Y** KEYWORD=ACTIVE) **ENTONCES**

INITIALIZE KEYWORD=k0 ó nodo inicial del hipercubo

Paso 5,6:

SI (KEYWORD=NEW **Y** k0 actualmente presente) **ENTONCES**

LINK NEW to k0

Set LINK=0

Set NEW=1

Paso 7:

KEYWORD=NEW **Y** k0,k1 actualmente presentes

a.{

LINK NEW to k1 //Se asume NEW conexiones k1. Si NEW

//conecta k0 solo las etiquetas cambiarán

//pero el proceso será el mismo

Set LINK=1

Set NEW=k2

LINK k2,k0 como {0,1}

}

b.MODIFY organización de KEYWORDS desde triángulo a cuadrado

{

COPY k0

LINK k0,k0=1

LINK k0,k2=0

}

Paso 8:

KEYWORD=NEW **Y** k0,k1,k2 actualmente presentes

{

Reemplazar COPY k0=NEW

NEW=k3

LINK k0,k3=1

LINK k3,k2=0

}

Paso 9:

KEYWORD=NEW **Y** k0,k1,k2,k3 actualmente presentes

a.{

LINK NEW to k0 //Se asume NEW conexiones k0. Si NEW

//conecta a otros, las etiquetas cambiarán

//pero el proceso será el mismo

Set LINK=2

Set NEW=k4

}

b.MODIFY organización de KEYWORDS desde cuadrado a cubo

{

COPY k1,k2,k3

LINK k0,k1=0

LINK k0,k3=1

LINK k1,k2=1

LINK k3,k2=0

LINK k2,k2=2

LINK k3,k3=2

LINK k1,k1=2

LINK k3,k4=1

LINK k1,k4=0

}

Paso 10:

KEYWORD=NEW **Y** k0, k1,k2, k3, k4 actualmente presentes

{

Reemplazar COPY k1=NEW

NEW=k5

LINK k4, k5=0

LINK COPY k2, k5=1

LINK k1, k5=2

}

Paso 11,12:

SI (k0 es borrado) **ENTONCES** //Para mostrar la eliminación de

//una KEYWORD se ha escogido k0

COPY k4 en el lugar de k0

LINKs se mantienen

Paso 13:

KEYWORD=NEW **Y** k1,k2,k3,k4,k5 actualmente presentes

{

Reemplazar COPY k1=NEW

NEW=k6

LINK COPY k2, k6=0

LINK k4, k6=1

LINK k3, k6=2

}

Paso 14:

KEYWORD=NEW **Y** k1,k2,k3,k4,k5,k6 actualmente presentes

{

Reemplazar COPY k2=NEW

NEW=k7

LINK k6, k7=0

LINK k5, k7=1

LINK k2, k7=2

}

Paso 15:

KEYWORD=NEW **Y** k1,k2,k3,k4,k5,k6,k7 actualmente presentes

{

Reemplazar COPY k4=NEW

NEW=k8

LINK k1, k8=0

LINK k3, k8=1

LINK k4, k8=2

}

FIN

Veamos una explicación del algoritmo acompañado de su representación gráfica:

Figura 14. Paso 6 algoritmo de hipercubos

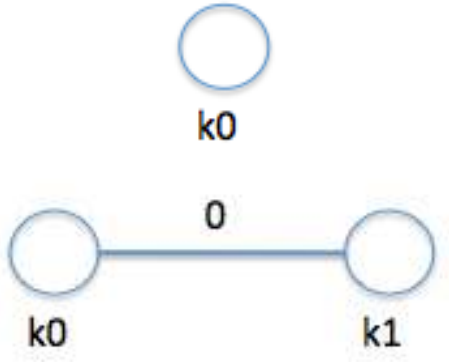


Figura 15. Paso 7a algoritmo de hipercubos

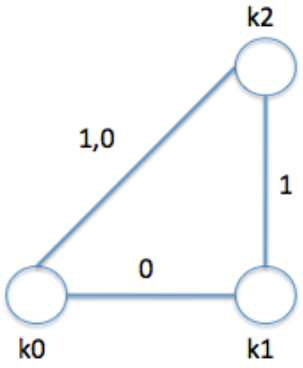


Figura 16. Paso 7b algoritmo de hipercubos

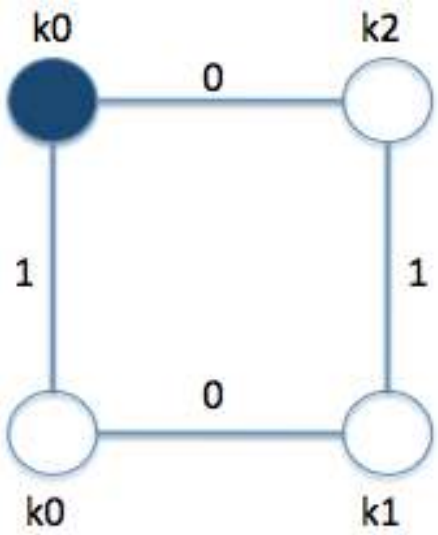
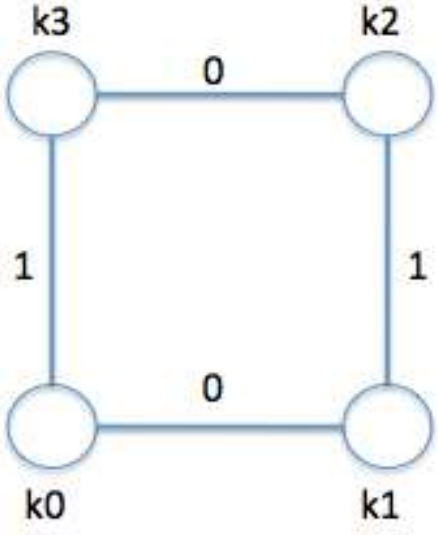
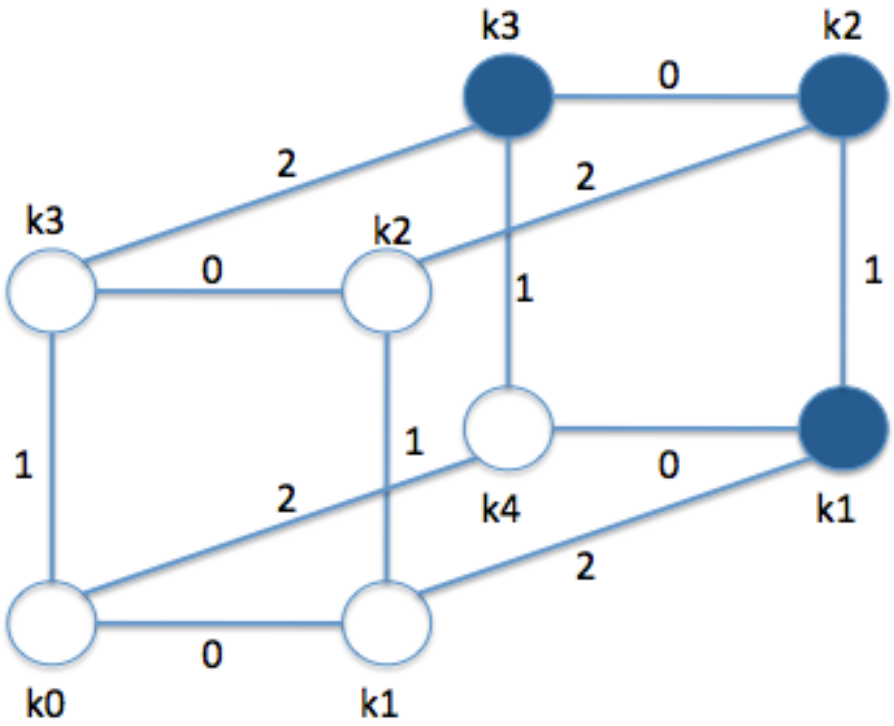


Figura 17. Paso 8 algoritmo de hipercubos



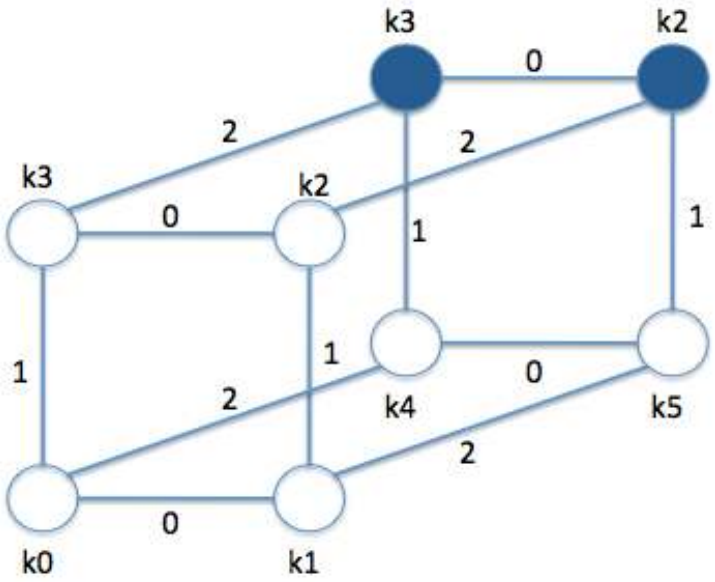
Las figuras 14,15,16 y 17 nos muestran desde el principio al paso número 8 del algoritmo anterior. Para todas las imágenes las palabras claves (KEYWORDS) son representados por círculos vacíos (blancos), las copias son representadas con círculos llenos (azul) y los números marcados en las líneas son la relación de vecindad o el enlace entre dos palabras claves.

Figura 18. Paso 9 algoritmo de hipercubos



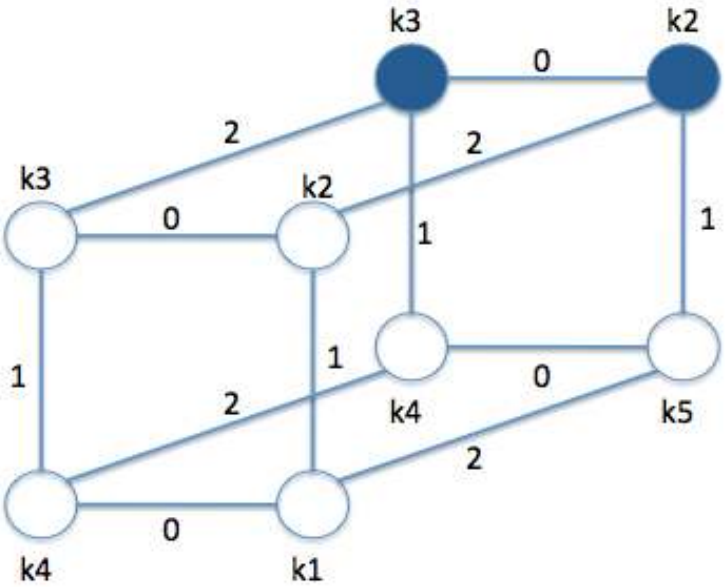
En la figura 18 se muestra el paso número 9 del algoritmo. Los círculos azules son copias de los originales, desarrollados para mantener la cohesión de la estructura.

Figura 19. Paso 10 algoritmo de hipercubos



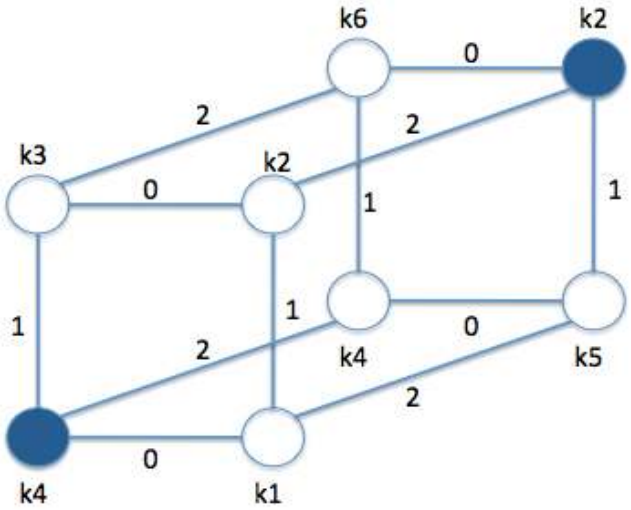
La figura 19 ilustra el paso número 10 del algoritmo.

Figura 20. Pasos 11 y 12 algoritmo de hipercubos



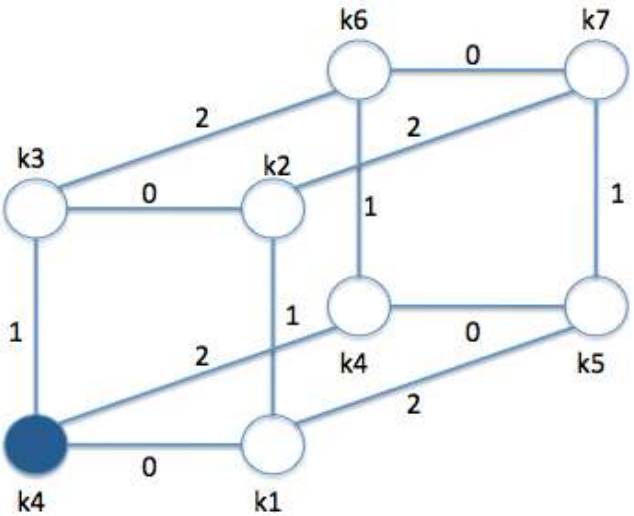
La figura 20 nos muestra los pasos 11 y 12 del algoritmo. Adicionalmente nos muestra la eliminación de una palabra clave dentro de la estructura.

Figura 21. Paso 13 algoritmo de hipercubos



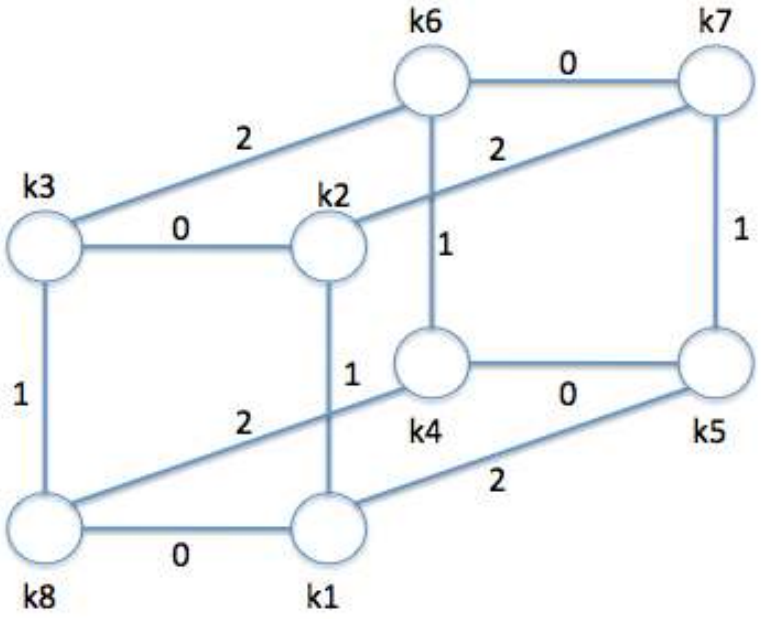
La figura 21 muestra el paso 13 del algoritmo.

Figura 22. Paso 14 algoritmo de hipercubos



La figura 22 muestra el paso 14 del algoritmo.

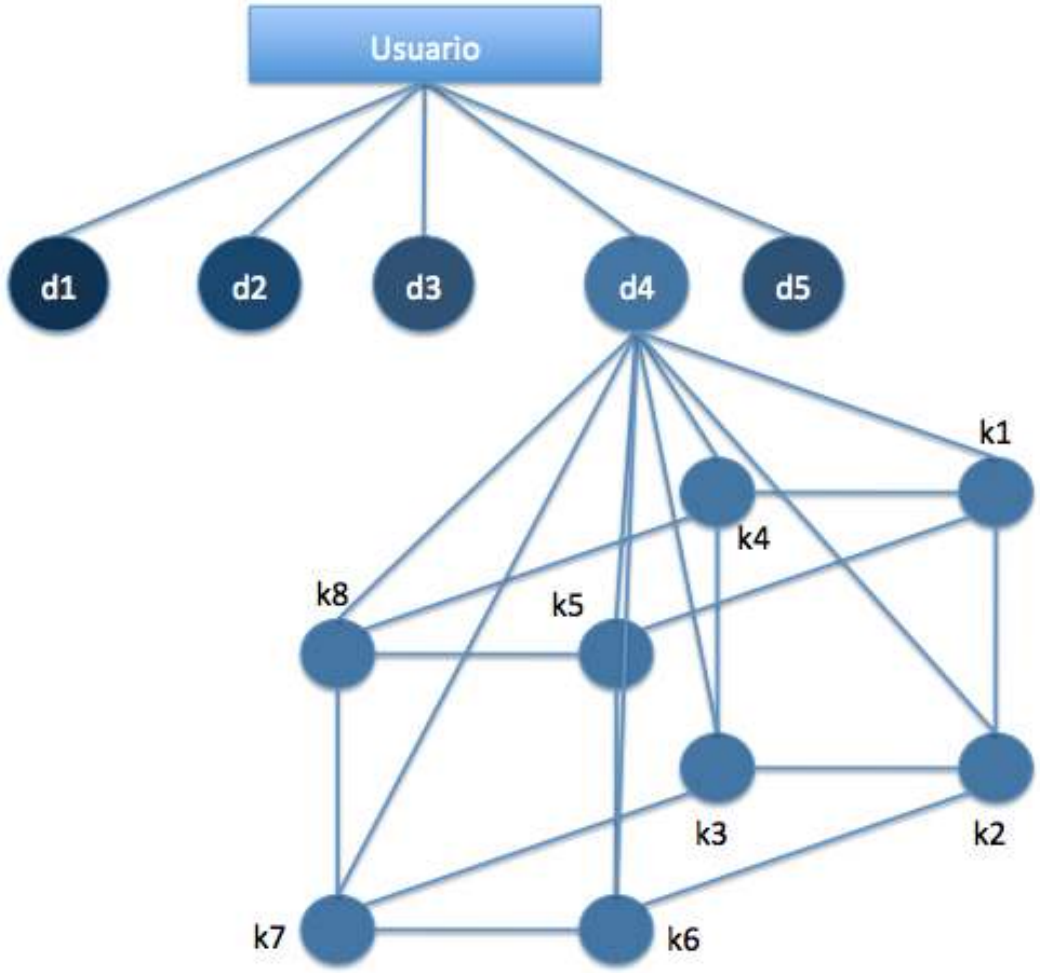
Figura 23. Paso final algoritmo de hipercubos



En la figura 23 se muestra el último paso del algoritmo y la posición final del cubo cuando todas las 8 palabras claves han sido incluidas. Esto lo llamamos estructura de hipercubo.

Después de aplicado el algoritmo se tiene la estructura de la ontología donde todas las palabras claves (KEYWORDS) de la ontología han sido organizadas en la estructura cúbica y el dominio tiene enlaces a todas las palabras claves. De este modo cuando un usuario selecciona un dominio después de especificar la palabra clave buscada será emitida a todas las palabras claves presentes en la ontología. A continuación se presenta la estructura de la ontología y las conexiones entre todas las palabras claves presentes en la ontología.

Figura 24. Estructura de dominios especificados por el usuario



En la gráfica anterior el usuario es representado como un rectángulo azul. Cuando especifica los dominios (d1,d2,d3,d4,d5) obtiene acceso a la ontología preparada para determinado dominio. La consulta es emitida hacia todos los nodos de la ontología. Cuando la información especificada es encontrada, la URL anexada es enviada para su respectiva descarga.

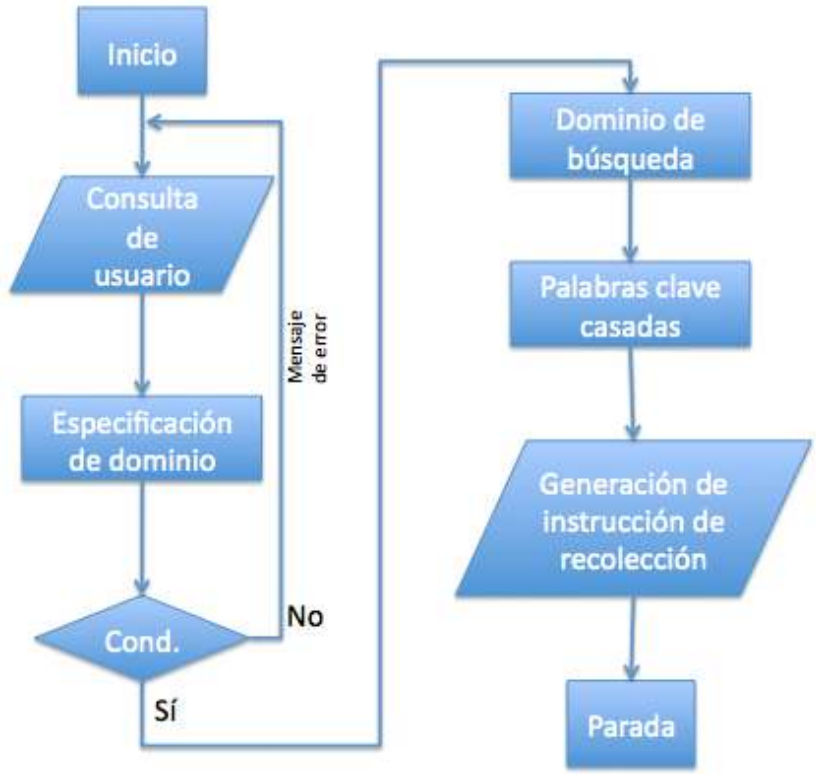
Luego de tener la estructura desarrollada el siguiente paso es la búsqueda. Como la estructura de la ontología está muy bien organizada la búsqueda se convierte

en un procedimiento sencillo pero efectivo. La búsqueda debe incluir los siguientes pasos:

Paso 1: Obtener la especificación del dominio de los usuarios.
Paso 2: Buscar para las palabras presentes en la consulta del usuario un dominio especificado por el usuario.
Paso 3: Notificar las palabras a todas las palabras claves de la ontología.
Paso 4: Devolver la URL anexada a las palabras claves las cuales casan con las palabras consultadas por el usuario.

A continuación se muestra gráficamente el proceso descrito para la búsqueda.

Figura 25. Proceso de búsqueda sobre una estructura de hipercubos



3.4.2. Acercamiento a un motor de búsqueda semántico basado en ontología[83]

A continuación se describe la arquitectura de un motor de búsqueda basado en ontologías. Para este caso nos basamos en un framework que se divide en cuatro módulos. Se explicará el módulo de procesamiento de ontologías con más profundidad.

Módulo de procesamiento de ontología. Principalmente establece las ontologías y ofrece interfaces para las ontologías en cualquier parte del sistema. Incluye un banco de ontologías que provee todas las ontologías Cuenta con tres submódulos:

Submódulo de marcado de ontologías: Su principal función es marcar sitios Web. Se compone de seis partes:

- Partición de palabras: Se encarga de partir las palabras en un texto.
- Identificación de conceptos: identifica los conceptos que vienen de las palabras anteriores.
- Vector de retiro de vocabulario: Retira conceptos de páginas Web para obtener el vector Eigen[84].
- Marca: Su tarea es encontrar la ontología en un banco de ontologías por medio de una interfaz Jena, a la luz del vocabulario encontrado con anterioridad.
- Cálculo del valor del peso: Su tarea es calcular el peso de cada concepto de acuerdo a su frecuencia y posición.

[83] Qi Yong; Hao Peijie; Hou Yu y Qiao Ya-nan. "The Research and Design of the Semantic Search Engine Based on Ontology". Proceedings of the Third International Conference on Semantics, Knowledge and Grid, 2007, p. 610-611

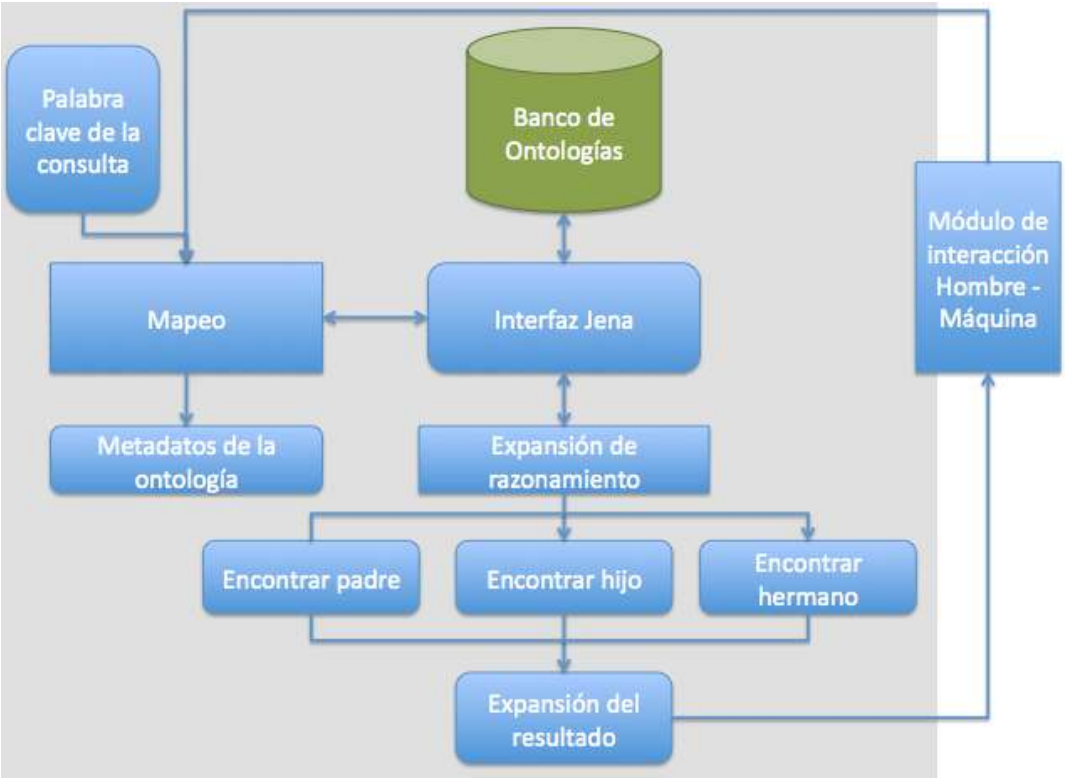
[84] "Determinants, Eigen Vectors". Disponible en: http://www.mathreference.com/la-det,eigen.html. Consultado: Abril 2010.

- Pesado: Encontrar los datos de peso en los metadatos de la página Web y traerlos a los metadatos locales para poder hacer ranking en el futuro.

Submódulo deductivo de expansión de ontologías: Su principal función es expandir las consultas de las palabras claves en consultas de expresiones ontológicas de metadatos por medio de la interacción con los usuarios. Se compone principalmente de dos partes:

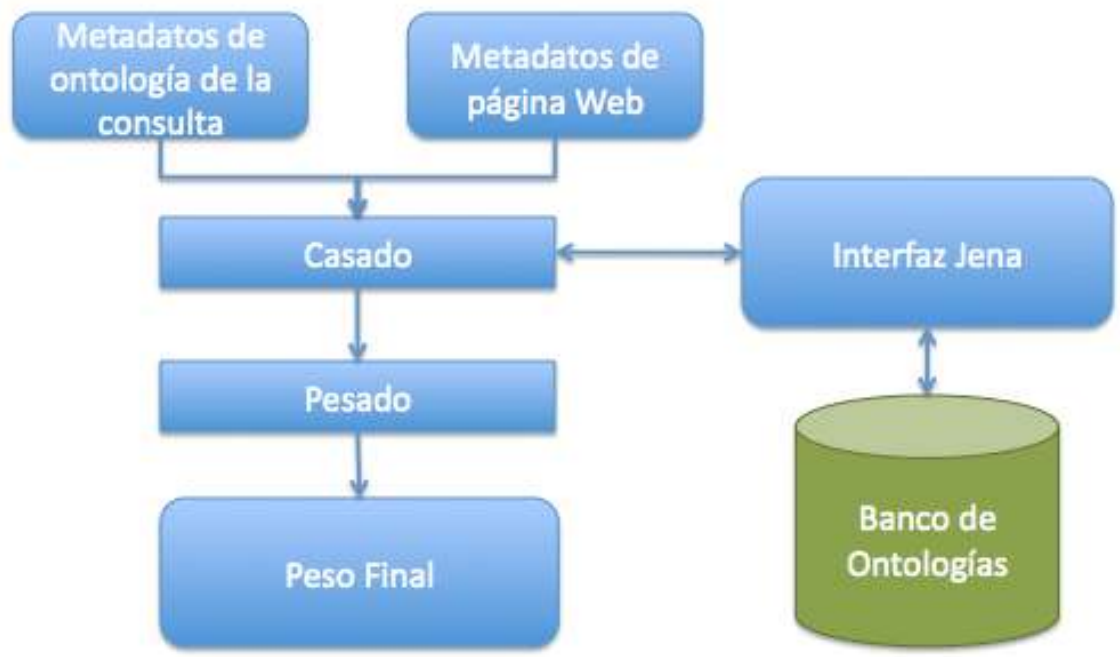
- Deducción y expansión: Deducir y expandir las ontologías mapeadas y mostrarlas al usuario.
- Mapeo: Mapear palabras claves para generar consultas de expresiones ontológicas por medio de una interfaz de operaciones ontológicas y una vía de expansión seleccionada por el usuario.

Figura 26. Arquitectura del sub-modulo deductivo de expansión de ontologías de un motor de búsqueda semántico basado en ontología



Submódulo de casado de ontologías: Su función es traer las ontologías de acuerdo al ranking para los resultados preliminares de la consulta.

Figura 27. Arquitectura del sub-modulo de casado de ontologías de un motor de búsqueda semántico basado en ontología



Módulo de colección de páginas Web. Colecciona información de páginas Web en términos de estrategias de recolección (crawling) y almacena los datos en la base de datos para operaciones posteriores.

Módulo de preprocesamiento. Analiza las páginas Web que han sido recolectadas y marca la ontología para cada sitio Web, también establece el índice invertido en los metadatos marcados.

Módulo de procesamiento de consultas. Provee a los usuarios interfaces para ejecutar las consultas y cargar los resultados.

3.4.3. Un motor de búsqueda ontológica basado en análisis semántico[85]

Se han visto varios motores de búsqueda a lo largo de este estado del arte, creemos que es necesario continuar con la revisión de otros motores para hacer un análisis más completo en el tema. A continuación se muestra el prototipo de un sistema llamado WI OntoSearch, el cual es un motor de búsqueda ontológica basado en algoritmos de vectores de pesos de conceptos (CWVMA, Concepts-weights vector matching algorithm).

Se definirá primero el algoritmo y luego la arquitectura del sistema WI OntoSearch.

CWVMA (Concepts-weights vector matching algorithm). Primero veamos algunas definiciones:

- Definición 1 (Peso del vector): Una ontología de n conceptos es mapeada en un vector $(r_1, r_2, ..., r_n)$ por regla de casado, $r_i \in [0,1]$. El valor de r_i denota la influencia de $i-esimo$ concepto dentro de una ontología y es decidido por la regla de casado. El vector $(r_1, r_2, ..., r_n)$ es llamado el vector de peso.
- Definición 2 (Vector peso-concepto): El conjunto de conceptos $(C_1, C_2, ..., C_n)$, donde C_i es el $i-esimo$ concepto de la ontología, y el correspondiente vector de peso $(r_1, r_2, ..., r_n)$ son llamados vector peso-concepto.
- Definición 3 (Ontología de comparación): Dada una ontología, es el estándar por el cual otra ontología puede ser medida o juzgada.
- Definición 4 (Ontología de evaluación): La ontología que necesita ser comparada con la ontología de comparación.

[85] Mingxia Gao, Chunnian Liu, Furong Chen. "An Ontology Search Engine Based on Semantic Analysis". Proceedings of the Third International Conference on Information Technology and Applications - Volume 02, pp. 256 - 259, 2005.

La idea básica del algoritmo es estimar la similitud semántica entre el mensaje de entrada y los resultados basados en palabras claves preliminares, esto es lo que definimos antes como ontología de comparación y ontología de evaluación. A continuación se presenta el algoritmo CWVMA:

Entrada: ontología de comparación: ontoComp, ontología de
evaluación ontoEval.

Salida: vector de resultados $(R_1, R_2, ..., R_m)$

COMIENZO

Parsear ontoComp y ontoEval dentro de $(I_1, I_2, ..., I_m)$ y $(C_1, C_2, ..., C_n)$;

Crear los correspondientes vectores de peso $(t_1, t_2, ..., t_m)$ y $(r_1, r_2, ..., r_n)$
por reglas de casado.

PARA todos los I_i en $(I_1, I_2, ..., I_m)$ haga:

Compare I_i con C_j en $(C_1, C_2, ..., C_n)$;

SI $I_i = C_j$ **ENTONCES**

$R_i = t_i \times r_j$

SINO

$R_i = 0$

FIN

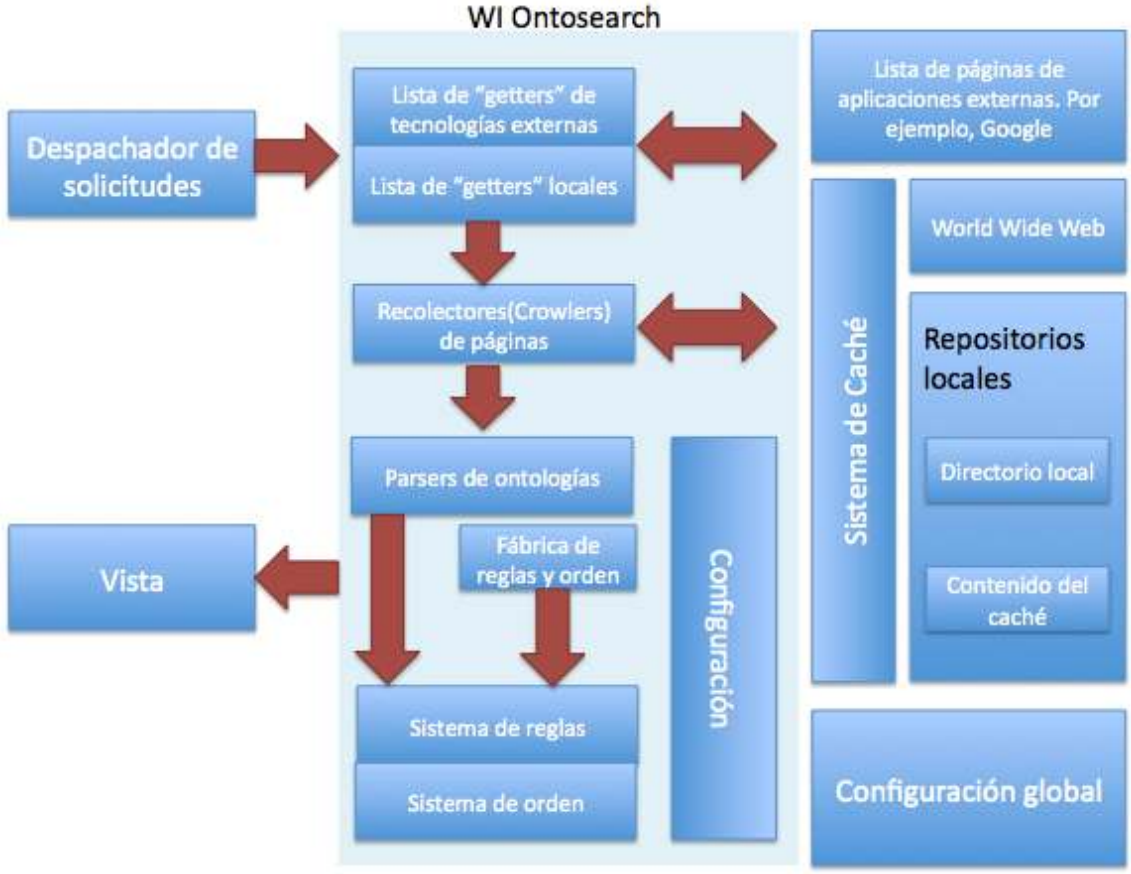
retornar vector de resultados $(R_1, R_2, ..., R_m)$.

FIN

R_i en el vector de resultados $(R_1, R_2, ..., R_m)$ puede expresar similitud semántica entre conceptos de la ontología de comparación y la ontología de evaluación. La $\sum_m R_i$ puede denotar la similitud entre la ontología de comparación y la ontología de evaluación. La complejidad del algoritmo es $O(m \times n)$, donde m es el número de conceptos de entrada y n es el número de conceptos en la ontología de evaluación.

Arquitectura de OntoSearch.

Figura 28. Arquitectura de OntoSearch



Es una arquitectura basada en tres capas. La primera ofrece un despachador de solicitudes y la vista del motor. La segunda (motor de búsqueda) cumple con la recuperación, filtrado y ordenado de resultados, también se encarga de dar calificaciones de ranking de acuerdo a diversos algoritmos. La tercera capa ofrece los datos y el caché.

- Definición 5 (Valor de ranking): Este es valor numérico que determina el ranking de la ontología de evaluación en todos los resultados. El sistema se

encarga de hacer el ranking de acuerdo a valores, filtros y orden de palabras claves. Adicionalmente, WI OntoSearch muestra la estructura semántica de la ontología de resultado para ayudar al usuario a conocer la estructura de la ontología, la semántica y dominio de aplicación.

3.4.4. Búsqueda de información educativa usando ontologías[86]

Cuando se busca información con fines educativos como lo es este estado del arte se encuentra que las palabras claves no son suficientes para buscar información en la Web y es cuando se necesita el uso de ontologías. En este caso se ha desarrollado una ontología con fines educativos para EDUNET[87] el sistema más grande de búsqueda de contenidos educativos en Corea.

"EDUNET es un servicio de información educativa que permite a todos los usuarios de los servicios educativos responder activamente a los rápidos cambios que tienen lugar en la información actual y en la sociedad del conocimiento. Es el portal más grande de educación, y apoya la distribución y la utilización de una amplia gama de contenidos educativos de alta calidad, a cualquier lugar y en cualquier momento"[88].

El sistema provee una búsqueda basada en palabras claves con una alta organización de metadatos que ha sido desarrollada y extendida basada en LOM[89]. Sin embargo el sistema se queda corto debido a los siguientes aspectos:

- Búsqueda poco sofisticada debido a la falta de relaciones entre las propiedades de los metadatos.

---

[86] Byoungchol Chang1, Dall-ho Ham1, Dae-sung Moon1, Yong S Choi2, Jaehyuk Cha1. "Educational Information Search Service Using Ontology". Seventh IEEE International Conference on Advanced Learning Technologies, 2007.
[87] "EDUNET". Disponible en: http://www.edunet4u.net/ Consultado: Abril 2010.
[88] Ibid.
[89] Hodgins, Wayne. "Learning Object Metadata". Disponible en: http://ltsc.ieee.org/wg12/ Consultado: Abril 2010.

- Ineficiencia en la búsqueda debido a la falta de semántica en las palabras claves.

La primera parte del trabajo se encargó de hacer un "binding" entre lo que ya existía en LOM a OWL, todo esto de la siguiente forma:

- Se condujo una encuesta al usuario y se hizo una pregunta de competencia para hacer "binding" de LOM a OWL.
- Los recursos de aprendizaje se definieron en la clase RecursosDeAprendizaje y son presentados como una instancia de esa clase.
- Se representó todo el contenido en LOM en OWL-DL permitiendo una semántica enriquecida gracias a la expresividad del lenguaje.

Construcción de la ontología de dominio. Para probar como la búsqueda semántica puede mejorar los resultados, se experimentó con la estructura semántica de "figuras geométricas" para octavo grado. Ésta se construyó con la colaboración de los contenidos del curso, profesores del curso y matemáticos. Para lograr una integración de la ontología de dominio y LOM, se usaron namespaces. Si la relación entre clases o propiedades dentro de la ontología de dominio son semánticamente a la ontología LOM, se combinan con propiedades OWL como owl:SameAs y owl:equivalentClasses.

Implementación y resultados. Se creó una interfaz personalizada de consultas basada en SPARQL. Se seleccionaron 2300 conceptos en el área de matemáticas para escuela secundaria y se clasificaron en ontologías con la ayuda de matemáticos. Entonces los metadatos fueron implementados dentro de las instancias de la clase de la ontología de dominio. Después de las pruebas se mostró que los resultados arrojados por la consulta con ontologías eran más específicos y precisos.